METplus

METplus User's Guide

version 5.1.0

May 16, 2024

Contents

Foreword: A note to METplus Wrappers users

This User's Guide is provided as an aid to users of the Model Evaluation Tools (MET) and it's companion package METplus Wrappers. MET is a suite of verification tools developed and supported to community via the Developmental Testbed Center (DTC) for use by the numerical weather prediction community. METplus Wrappers are intended to be a suite of Python wrappers and ancillary scripts to enhance the user's ability to quickly set-up and run MET. Over the next year, METplus Wrappers will become the authoritative repository for verification of the Unified Forecast System.

It is important to note here that METplus Wrappers is an evolving software package. The METplus Wrappers package was first released in 2017. This documentation describes the 5.1.0 release (July 2023). Intermediate releases may include bug fixes. METplus Wrappers is also be able to accept new modules contributed by the community. While we are setting up our community contribution protocol, please create a post in the METplus GitHub Discussions Forum and inform us of your desired contribution. We will then determine the maturity of any new verification method and coordinate the inclusion of the new module in a future version.

Citations

The citation for this User's Guide should be:

Opatz, J., J. Halley Gotway, T. Jensen, J. Vigh, M. Row, C. Kalb, H. Fisher, L. Goodrich, D. Adriaansen, M. Win-Gildenmeister, G. McCabe, J. Prestopnik, J. Frimel, L. Blank, T. Arbetter, 2023: The METplus Version 5.1.0 User's Guide. Developmental Testbed Center. Available at: https://github.com/dtcenter/METplus/releases.

Acknowledgments

We thank all of the METplus sponsors including: DTC partners (NOAA, NCAR, USAF, and NSF), along with NOAA/Office of Atmospheric Research (OAR), NOAA/National Weather Service, NOAA/Joint Technology Transfer Program (JTTI), NOAA/Subseasonal to Seasonal (S2S) Project, NOAA/Unified Forecast System Research to Operations Project (UFS R2O), Met Office, and the Naval Research Laboratory (NRL). Thanks also go to the staff at the Developmental Testbed Center for their help, advice, and many types of support. We released METplus Alpha in February 2017 and would not have made a decade of cutting-edge verification support without those who participated in DTC planning workshops and the United Forecast System Working Groups (UFS WGs). Finally, the National Center for Atmospheric Research (NCAR) is sponsored by NSF.

Chapter 1

Overview

1.1 Purpose and organization of the User's Guide

The goal of this User's Guide is to equip users with the information needed to use the Model Evaluation Tools (MET) and its companion package METplus Wrappers. MET is a set of verification tools developed and supported to community via the Developmental Testbed Center (DTC) for use by the numerical weather prediction community. METplus Wrappers is a suite of Python wrappers and ancillary scripts to enhance the user's ability to quickly set-up and run MET. Over the next few years, METplus Wrappers will become the authoritative repository for verification of the Unified Forecast System.

The METplus Wrappers User's Guide is organized as follows. An overview of METplus Wrappers can be found below. *Installation* (page 29) contains basic information about how to get started with METplus Wrappers - including system requirements, required software, and how to download METplus Wrappers. *Configuration* (page 37) provides information about configuring your environment and METplus Wrappers installation.

1.2 The Developmental Testbed Center (DTC)

METplus Wrappers has been developed, and will be maintained and enhanced, by the Developmental Testbed Center (DTC; http://www.dtcenter.org/). The main goal of the DTC is to serve as a bridge between operations and research and to facilitate the activities of these two important components of the numerical weather prediction (NWP) community. The DTC provides an environment that is functionally equivalent to the operational environment in which the research community can test model enhancements; the operational community benefits from DTC testing and evaluation of models before new models are implemented operationally. METplus Wrappers serves both the research and operational communities in this way - offering capabilities for researchers to test their own enhancements to models and providing a capability for the DTC to evaluate the strengths and weaknesses of advances in NWP prior to operational implementation.

METplus Wrappers will also be available to DTC visitors and the NOAA Unified Forecast System (UFS) and NCAR System for Integrated Modeling of the Atmosphere (SIMA) modeling communities for testing and evaluation of new model capabilities, applications in new environments, and so on. The METplus Wrappers release schedule is coincident with the MET release schedule and the METplus Wrappers major release number is six less than the MET major release number (e.g. MET 8.X is released with METplus Wrappers 2.X).

1.3 METplus Wrappers goals and design philosophy

METplus Wrappers is a Python scripting infrastructure for the MET tools. The primary goal of METplus Wrappers development is to provide MET users with a highly configurable and simple means to perform model verification using the MET tools. Prior to the availability of METplus Wrappers, users who had more complex verifications that required the use of more than one MET tool were faced with setting up multiple MET config files and creating some automation scripts to perform the verification. METplus Wrappers provides the user with the infrastructure to modularly create the necessary steps to perform such verifications.

METplus Wrappers has been designed to be modular and adaptable. This is accomplished through wrapping the MET tools with Python and the use of hierarchical configuration files to enable users to readily customize their verification environments. Wrappers can be run individually, or as a group of wrappers that represent a sequence of MET processes. New wrappers can readily be added to the METplus Wrappers package due to this modular design. Currently, METplus Wrappers can easily be applied by any user on their own computer platform that supports Python 3.8. We have deprecated support to Python 2.7.

The METplus Wrappers code and documentation is maintained by the DTC in Boulder, Colorado. METplus Wrappers is freely available to the modeling, verification, and operational communities, including universities, governments, the private sector, and operational modeling and prediction centers through a publicly accessible GitHub repository. Refer to *Getting the METplus Wrappers source code* (page 31) for simple examples of obtaining METplus Wrappers.

1.4 METplus Wrappers Components

The major components of the METplus Wrappers package are METplus Python wrappers to the MET tools, MET configuration files and a hierarchy of METplus Wrappers configuration files. Some Python wrappers do not correspond to a particular MET tool, but wrap utilities to extend METplus functionality.

1.5 METplus Components Python Requirements

Name	Version	METplus Component	Source
Python 3.10.4+		METplus wrappers, METcalcpy, METplotpy, METdataio	
cartopy	>=0.21.1	METplus wrappers, METcalcpy, METplotpy	https://scitools.org.uk/ca
cfgrib		METplus wrappers	https://pypi.org/project/c
cmocean		METcalcpy, METplotpy	https://pypi.org/project/c
dateutil	>=2.8.2	METplus wrappers	https://github.com/dateu
eofs		METplus wrappers, METcalcpy, METplotpy	https://pypi.org/project/e
h5py		METplus wrappers	https://github.com/h5py/
imutils	>=0.5.4	METplotpy	https://pypi.org/project/i
imageio		METcalcpy, METplotpy	https://pypi.org/project/i
lxml	>=4.9.1	METcalcpy, METplotpy, METdataio	https://pypi.org/project/l
matplotlib	>=3.6.3	METplus wrappers, METcalcpy, METplotpy	https://matplotlib.org/sta

Name	Version	METplus Component	Source
metcalcpy		METplus wrappers, METcalcpy, METplotpy	https://github.com/dtcen
metplotpy		METplus wrappers	https://github.com/dtcen
metpy	>=1.4.0	METplus wrappers	https://www.unidata.uca
nc-time-axis	1.4	METplotpy stratosphere_diagnostics	https://github.com/SciTo
netCDF4	>=1.6.2	METplus wrappers, METcalcpy, METplotpy	https://unidata.github.io/
numpy	>=1.24.2	METplus wrappers, METcalcpy, METplotpy, METdataio	https://numpy.org/
pandas	>=1.5.2	METplus wrappers, METcalcpy, METplotpy, METdataio	https://pypi.org/project/
pint	>=0.20.1	METcalcpy	https://github.com/hgrec
plotly	>=5.13.0	METcalcpy, METplotpy	https://github.com/plotly
pygrib		METplus wrappers	https://github.com/jswhi
pylab		METplus wrappers	https://pypi.org/project/
pymysql	>=1.0.2	METcalcpy, METplotpy, METdataio	https://pypi.org/project/
pyproj	>=2.3.1	METplus wrappers	https://github.com/pypro
pyresample		METplus wrappers	https://github.com/pytro
pytest	>=7.2.1	METcalcpy, METplotpy, METdataio	https://github.com/pytes
python-kaleido	>=0.2.1	METcalcpy, METplotpy	https://pypi.org/project/
pyyaml	>=6.0	METcalcpy, METplotpy, METdataio	https://github.com/yaml/
scikit-image	>=0.19.3	METcalcpy, METplotpy	https://scikit-image.org
scikit-learn	>=1.2.2	METplus wrappers, METcalcpy, METplotpy	https://github.com/scikit
scipy	>=1.9.3	METplus wrappers, METcalcpy, METplotpy	https://www.scipy.org/
sklearn		METplus wrappers	https://www.kite.com/py
xarray	>=2023.1.0	METplus wrappers, METcalcpy, METplotpy	https://xarray.pydata.org/
xesmf		METplus wrappers	NOTE: The xesmf package
yaml		METcalcpy, METplotpy	https://pypi.org/project/

1.6 Future development plans

METplus Wrappers is an evolving application. New capabilities are planned in controlled, successive version releases that are synchronized with MET releases. Software bugs and user-identified problems will be documented using GitHub issues and fixed either in the next bugfix or official release. Future METplus Wrappers development plans are based on several contributing factors, including the needs of both the operational and research community. Issues that are in the development queue detailed in the "Issues" section of the GitHub repository. Please create a post in the METplus GitHub Discussions Forum with any questions.

1.7 Code support

Support for METplus Wrappers is provided through the METplus GitHub Discussions Forum. We will endeavor to respond to requests for help in a timely fashion. In addition, information about METplus Wrappers and tools that can be used with MET are provided on the MET Users web page.

We welcome comments and suggestions for improvements to METplus Wrappers, especially information regarding errors. Comments may be submitted using the MET Feedback form available on the MET website. In addition, comments on this document would be greatly appreciated. While we cannot promise to incorporate all suggested changes, we will certainly take all suggestions into consideration.

METplus Wrappers is a "living" set of wrappers and configuration files. Our goal is to continually enhance it and add to its capabilities. Because our time, resources, and talents can at times be limited, we welcome contributed code for future versions of METplus. These contributions may represent new use cases or new plotting functions. For more information on contributing code to METplus Wrappers, please create a post in the METplus GitHub Discussions Forum.

Chapter 2

METplus Release Information

Users can view the releaseCycleStages section of the Release Guide for descriptions of the development releases (including beta releases and release candidates), official releases, and bugfix releases for the METplus Components.

2.1 METplus Components Release Note Links

- MET (latest, development)
- METviewer (latest, development)
- METplotpy (latest, development)
- METcalcpy (latest, development)
- METdataio (latest, development)
- METexpress (latest, development)
- METplus Wrappers (latest, *upgrade instructions* (page 13), development)

2.2 METplus Wrappers Release Notes

When applicable, release notes are followed by the GitHub issue number which describes the bugfix, enhancement, or new feature.

2.2.1 METplus Version 5.1.0 Release Notes (2023-07-31)

Enhancements

- Add support for multiple interp widths (#2049)
- TCPairs - Add support for setting consensus.write_members (#2054)
- Update use cases to use new Python directory structure in MET (#2115)
- Add support for new multivariate MODE settings (#2197, #2210, #2230, #2235)

Bugfix

- StatAnalysis - allow run once for each valid time (#2026)
- App specific OBS_WINDOW variables not taking precedence over generic (#2006)
- Skip-if-output-exists logic incorrectly skips files (#2096)
- PointStat -obs_valid_beg/end arguments not set properly (#2137)
- Allow setting of convert, censor_thresh, and censor_val in regrid dictionary (#2082)
- TCPairs setting -diag option causes failure (#2179)
- Define the order of the forecast variables numerically rather than alphabetically (#2070)
- Prevent error if no commands were run because they were skipped (#2098)
- Allow spaces for complex categorical thresholds (#2189)
- PCPCombine - Extra field options not set in -subtract mode (#2161)
- StatAnalysis time shifting failure (#2168)
- TCPairs: skip times logic incorrectly skips additional times (#2212)
- TCStat fails if -out_stat directory does not exist (#2241)
- Ensure log instances for concurrent runs are unique (#2245)
- PointStat - Support running with no MASK provided (#1853)

New Wrappers

- TCDiag (beta) (#1626)

New Use Cases

- Multi-Variate MODE (#1516)
- Read in Argo profile data netCDF files for use in METplus with python embedding (#1977)
- PANDA-C: MPAS to SATCORPS (#2188)
- PANDA-C: MPAS to MERRA2 (#2188)
- PANDA-C: MPAS to ERA5 (#2188)
- PANDA-C: GFS to SATCORPS (#2188)
- PANDA-C: GFS to MERRA2 (#2188)
- PANDA-C: GFS to ERA5 (#2188)
- PointStat AMDAR PBLH with python embedding (#2198)
- Space-time cross-spectra for S2S forecasts (#2136)

Documentation

- Enhance the Release Notes by adding dropdown menus (#2076)
- Update the METplus Components Python Requirements (#1978, #2016)
- Add documentation on support for releases to the Release Guide (#2106)
- Add use case quick search keywords for METplotpy and METcalcpy (#2151)

Internal

- Improve use case testing (#685)
- Update conda environments to use 3.10 for automated use case tests (#2005)
- Add modulefiles to the repository (#2015)
- **Upgrade to using Python 3.10.4** (#2022)
- Add 'LICENSE.md' to the METplus repo (#2058)
- Update Contributor's Guide to use GH Action to update truth data (#2068)
- Enhance GitHub Workflow documentation (#2147)
- Update the development release guide instructions to remove references to a Coordinated release (#2159)
- Refactored code to resolve many SonarQube items (#1610)
- Improve Contributor's Guide (#2138, #2207)
- Bugfix: Fix difference test logic (#2244)
- Remove base environment from Docker Conda images (#2249)

- PR Templates (all METplus component repos): Improve language about linking relevant issue(s) (#2257)

2.2.2 METplus Version 5.0.0 Release Notes (2022-12-09)

Warning: MAJOR CHANGES:

- The LOOP_ORDER config variable was removed. The variable set in a user's config file will be ignored in favor of executing the logic that corresponds to *LOOP_ORDER = processes*, where all times are processed for the first item in the PROCESS_LIST, then all times are processed for the second item in the PROCESS_LIST, etc. This may change the order that commands are executed in a use case, but it should not change the results.
- The METplus Dockerfile was moved to internal/scripts/docker. It was previously found in scripts/docker.
- Use cases that include **EnsembleStat** wrapper will require config file updates. See *METplus Wrappers Upgrade Instructions* (page 13).
- The default value of *SCRUB_STAGING_DIR* is now *True*. This means some intermediate files that are auto-generated by METplus such as file lists and uncompressed files will automatically be removed unless this option is set by the user. These files are typically only used to debug unexpected issues.
- The default value of *METPLUS_CONF* now includes the *LOG_TIMESTAMP* so each METplus run will generate a unique final config file, e.g. metplus_final.conf.20220921121733.

Enhancements

- **Enhance MODE wrapper to support multi-variate MODE** (#1585)
- **Allow FCST_IS_PROB variable setting specific to tool (FCST_<tool_name>_IS_PROB)** (#1586)
- **Enhance climatology field settings to be consistent with fcst/obs field** (#1599)
- Update Hovmoeller Use case to use updated Hovmoeller plotting (#1650)
- **Update the EnsembleStat wrapper and use case examples to remove ensemble post processing logic** (#1816)
- Enhance logic to consistently create directories (#1657)
- Create checksum for released code (#262)
- Add the user ID to the log output at beginning and end of each METplus wrappers run (dtcenter/METplus-Internal#20)
- Update logic to name final conf and intermediate files with a unique identifier (dtcenter/METplus-Internal#32)

- Change default logging time information (dtcenter/METplus-Internal#34)
- **Remove LOOP_ORDER config variable** (#1687)
- **Add unique identifier for each METplus run to configuration** (#1829)
- StatAnalysis - Support setting multiple jobs (#1842)
- StatAnalysis - Set MET verbosity (#1772)
- StatAnalysis - Support using both init/valid variables in string substitution (#1861)
- StatAnalysis - Allow filename template tags in jobs (#1862)
- StatAnalysis - Support looping over groups of list items (#1870)
- StatAnalysis - Allow processing of time ranges other than daily (#1871)
- StatAnalysis - Add support for using a custom loop list (#1893)
- Remove MakePlots wrapper (#1843)
- Add support in EnsembleStat wrapper for setting -ens_mean command line argument (#1569)
- Enhance METplus to have better signal handling for shutdown events (dtcenter/METplus-Internal#27)
- TCPairs and TCStat - add support for new config options and command line arguments (#1898)
- Enhance the GridStat and PointStat wrappers to handle the addition of SEEPS (#1953)
- SeriesAnalysis - add support for setting mask dictionary (#1926)
- Update Python requirement to 3.8.6 (#1566)
- Enhance StatAnalysis wrapper to support now and today (#1669)
- **Clean up use case configuration files** (#1402)
- Add support for creating multiple input datasets (#1694)

Bugfixes

- PCPCombine - custom loop list does not work for subtract method (#1884)
- Set level properly in filename template for EnsembleStat forecast input (#1910)
- Prevent duplicate observation files using a file window if compressed equivalent files exist in same directory (#1939)
- Allow NA value for <TOOL-NAME>_CLIMO_[MEAN/STDEV]_HOUR_INTERVAL (#1787)
- Reconcile setting of METPLOTPY_BASE for use cases (#1713)
- Add support for the {custom} loop string in the MODEL config variable (#1382)
- Fix PCPCombine extra options removal of semi-colon (#1534)

- Fix reset of arguments for some wrappers (i.e. GenEnsProd) after each run (#1555)
- Enhance METDbLoad Wrapper to find MODE .txt files (#1608)
- Add missing brackets around list variable values for StatAnalysis wrapper (#1641)
- Allow NA value for <TOOL-NAME>_CLIMO_[MEAN/STDEV]_DAY_INTERVAL (#1653)

New Wrappers

- PlotPointObs (#1489)

New Use Cases

- PANDA-C use cases (#1686)
- MJO-ENSO diagnostics (#1330)
- Probability of Exceedence for 85th percentile temperatures (#1808)
- FV3 Physics Tendency plotting via METplotpy (#1852)
- StatAnalysis Python Embedding using IODA v2.0 (#1453)
- StatAnalysis Python Embedding to read native grid (u-grid) (#1561)

Documentation

- Update documentation to include instructions to disable UserScript wrapper (dtcenter/METplus-Internal#33)

Internal

- Organize utility scripts used by multiple wrappers (#344)
- Fix GitHub Actions warnings - update the version of actions and replace set-output (#1863)
- Update diff logic to handle CSV files that have rounding differences (#1865)
- Add unit tests for expected failure (dtcenter/METplus-Internal#24)
- Add instructions in Release Guide for "Recreate an Existing Release" (#1746)
- Add modulefiles used for installations on various machines (#1749)
- Document GitHub Discussions procedure for the Contributor's Guide (#1159)
- Create a METplus "Release Guide" describing how to build releases for the METplus components (#673)
- Update documentation about viewing RTD URLs on branches (#1512)

2.3 METplus Wrappers Upgrade Instructions

2.3.1 EnsembleStat/GenEnsProd (v5.0.0)

Note: If *EnsembleStat* (page 101) is not found in the *PROCESS_LIST* for any use cases, then this section is not relevant. If upgrading from v5.0.X to v5.1.0, then this section is not relevant.

The METplus v5.0.0 coordinated release includes changes that remove ensemble product generation from EnsembleStat. GenEnsProd is now required to generate ensemble products. There are 3 situations listed below that require slightly different modifications:

- *Case 1: EnsembleStat only generating ensemble products* (page 13)
- *Case 2: EnsembleStat performing ensemble verification but not generating ensemble products* (page 16)
- *Case 3: EnsembleStat generating ensemble products and performing ensemble verification* (page 17)

2.3.1.1 Case 1: EnsembleStat only generating ensemble products

If the use case had been calling EnsembleStat **WITHOUT** the **-grid_obs** or **-point_obs** command line options, we can assume it was only doing ensemble post-processing. That call to EnsembleStat should be replaced with a call to GenEnsProd instead.

2.3.1.1.1 Rename Variables

Old Name	New Name
FCST_ENSEMBLE_STAT_INPUT_DIR	GEN_ENS_PROD_INPUT_DIR
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE	GEN_ENS_PROD_INPUT_TEMPLATE
ENSEMBLE_STAT_OUTPUT_DIR	GEN_ENS_PROD_OUTPUT_DIR
ENSEMBLE_STAT_OUTPUT_TEMPLATE	GEN_ENS_PROD_OUTPUT_TEMPLATE **and add filename, see** *below* (page 14)
ENSEMBLE_STAT_N_MEMBERS	GEN_ENS_PROD_N_MEMBERS
ENSEMBLE_STAT_ENS_THRESH	GEN_ENS_PROD_ENS_THRESH
ENSEMBLE_STAT_ENS_VLD_THRESH	GEN_ENS_PROD_VLD_THRESH
ENSEMBLE_STAT_ENSEMBLE_FLAG_LATLON	GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN	GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN
ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV	GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV
ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS	GEN_ENS_PROD_ENSEMBLE_FLAG_MINUS
ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS	GEN_ENS_PROD_ENSEMBLE_FLAG_PLUS
ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN	GEN_ENS_PROD_ENSEMBLE_FLAG_MIN
ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX	GEN_ENS_PROD_ENSEMBLE_FLAG_MAX
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE	GEN_ENS_PROD_ENSEMBLE_FLAG_RANGE
ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT	GEN_ENS_PROD_ENSEMBLE_FLAG_VLD_COUNT

continues on next page

Table 1 – continued from previous page

Old Name	New Name
ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY	GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY
ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP	GEN_ENS_PROD_ENSEMBLE_FLAG_NEP
ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP	GEN_ENS_PROD_ENSEMBLE_FLAG_NMEP
ENSEMBLE_STAT_REGRID_TO_GRID	GEN_ENS_PROD_REGRID_TO_GRID
ENSEMBLE_STAT_REGRID_METHOD	GEN_ENS_PROD_REGRID_METHOD
ENSEMBLE_STAT_REGRID_WIDTH	GEN_ENS_PROD_REGRID_WIDTH
ENSEMBLE_STAT_REGRID_VLD_THRESH	GEN_ENS_PROD_REGRID_VLD_THRESH
ENSEMBLE_STAT_REGRID_SHAPE	GEN_ENS_PROD_REGRID_SHAPE
ENSEMBLE_STAT_NBRHD_PROB_WIDTH	GEN_ENS_PROD_NBRHD_PROB_WIDTH
ENSEMBLE_STAT_NBRHD_PROB_SHAPE	GEN_ENS_PROD_NBRHD_PROB_SHAPE
ENSEMBLE_STAT_NBRHD_PROB_VLD_THRESH	GEN_ENS_PROD_NBRHD_PROB_VLD_THRESH
ENSEMBLE_STAT_NMEP_SMOOTH_VLD_THRESH	GEN_ENS_PROD_NMEP_SMOOTH_VLD_THRESH
ENSEMBLE_STAT_NMEP_SMOOTH_SHAPE	GEN_ENS_PROD_NMEP_SMOOTH_SHAPE
ENSEMBLE_STAT_NMEP_SMOOTH_METHOD	GEN_ENS_PROD_NMEP_SMOOTH_METHOD
ENSEMBLE_STAT_NMEP_SMOOTH_WIDTH	GEN_ENS_PROD_NMEP_SMOOTH_WIDTH
ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_DX	GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_DX
ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_RADIUS	GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_RADIUS

2.3.1.1.2 Set GenEnsProd output template to include filename

- **If the EnsembleStat output template was set**, then copy the value and add a template for the NetCDF output filename at the end following a forward slash '/' character.

  For example, if

  ```
  ENSEMBLE_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}
  ```

  then set

  ```
  GEN_ENS_PROD_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}/gen_ens_prod_{valid?fmt=%Y%m%d_%H%M
  ↪%S}V_ens.nc
  ```

  or something similar.

- **If the EnsembleStat output template was not set,** then set GenEnsProd's template to the desired NetCDF output filename.

  Example:

  ```
  GEN_ENS_PROD_OUTPUT_TEMPLATE = gen_ens_prod_{valid?fmt=%Y%m%d_%H%M%S}V_ens.nc
  ```

Ensure that any downstream wrappers in the PROCESS_LIST are configured to read the correct GenEnsProd output file instead of the _ens.nc file that was previously generated by EnsembleStat.

2.3.1.1.3 Handle ENS_VAR<n> variables

If ENS_VAR<n>_* variables are already set, this section can be skipped.

If ENS_VAR<n>_* variables are not set, add ENS_VAR<n> variables.

- If FCST/OBS verification is **NOT** being performed in the use case using another wrapper, then rename the FCST_VAR<n> variables to ENS_VAR<n>.

  For example:

Old Name	New Name
FCST_VAR1_NAME	ENS_VAR1_NAME
FCST_VAR1_LEVELS	ENS_VAR1_LEVELS
FCST_VAR2_NAME	ENS_VAR2_NAME
FCST_VAR2_LEVELS	ENS_VAR2_LEVELS

- If FCST/OBS verification is being performed by another tool, then add ENS_VAR<n> variables and set them equal to the corresponding FCST_VAR<n> values.

  For example:

```
ENS_VAR1_NAME = {FCST_VAR1_NAME}
ENS_VAR1_LEVELS = {FCST_VAR1_LEVELS}
ENS_VAR2_NAME = {FCST_VAR2_NAME}
ENS_VAR2_LEVELS = {FCST_VAR2_LEVELS}
```

2.3.1.1.4 Remove Variables

Remove any remaining ENSEMBLE_STAT_* variables that are no longer used.

Some examples:

ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK
ENSEMBLE_STAT_ENSEMBLE_FLAG_WEIGHT
ENSEMBLE_STAT_MESSAGE_TYPE
ENSEMBLE_STAT_OUTPUT_FLAG_ECNT
ENSEMBLE_STAT_OUTPUT_FLAG_RPS
ENSEMBLE_STAT_OUTPUT_FLAG_RHIST
ENSEMBLE_STAT_OUTPUT_FLAG_PHIST
ENSEMBLE_STAT_OUTPUT_FLAG_ORANK
ENSEMBLE_STAT_OUTPUT_FLAG_SSVAR
ENSEMBLE_STAT_OUTPUT_FLAG_RELP
ENSEMBLE_STAT_OUTPUT_FLAG_PCT
ENSEMBLE_STAT_OUTPUT_FLAG_PSTD
ENSEMBLE_STAT_OUTPUT_FLAG_PJC
ENSEMBLE_STAT_OUTPUT_FLAG_PRC
ENSEMBLE_STAT_OUTPUT_FLAG_ECLV
ENSEMBLE_STAT_DUPLICATE_FLAG
ENSEMBLE_STAT_SKIP_CONST
ENSEMBLE_STAT_OBS_ERROR_FLAG
ENSEMBLE_STAT_ENS_SSVAR_BIN_SIZE
ENSEMBLE_STAT_ENS_PHIST_BIN_SIZE
ENSEMBLE_STAT_CI_ALPHA
ENSEMBLE_STAT_MASK_GRID
ENSEMBLE_STAT_MASK_POLY
ENSEMBLE_STAT_INTERP_FIELD
ENSEMBLE_STAT_INTERP_VLD_THRESH
ENSEMBLE_STAT_INTERP_SHAPE
ENSEMBLE_STAT_INTERP_METHOD
ENSEMBLE_STAT_INTERP_WIDTH
ENSEMBLE_STAT_OBS_QUALITY_INC/EXC
ENSEMBLE_STAT_GRID_WEIGHT_FLAG

2.3.1.2 Case 2: EnsembleStat performing ensemble verification but not generating ensemble products

The use case will no longer generate a **_ens.nc** file and may create other files (**_orank.nc** and **txt**) that contain requested output.

2.3.1.2.1 Rename Variables

Old Name	New Name
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN	ENSEMBLE_STAT_NC_ORANK_FLAG_MEAN
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK	ENSEMBLE_STAT_NC_ORANK_FLAG_RANK
ENSEMBLE_STAT_ENSEMBLE_FLAG_WEIGHT	ENSEMBLE_STAT_NC_ORANK_FLAG_WEIGHT
ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT	ENSEMBLE_STAT_NC_ORANK_FLAG_VLD_COUNT

2.3.1.2.2 Remove Variables

All ENS_VAR<n>_* variables
All ENSEMBLE_STAT_ENSEMBLE_FLAG_* variables
ENSEMBLE_STAT_NBRHD_PROB_WIDTH
ENSEMBLE_STAT_NBRHD_PROB_SHAPE
ENSEMBLE_STAT_NBRHD_PROB_VLD_THRESH
ENSEMBLE_STAT_NMEP_SMOOTH_VLD_THRESH
ENSEMBLE_STAT_NMEP_SMOOTH_SHAPE
ENSEMBLE_STAT_NMEP_SMOOTH_METHOD
ENSEMBLE_STAT_NMEP_SMOOTH_WIDTH
ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_DX
ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_RADIUS

2.3.1.3 Case 3: EnsembleStat generating ensemble products and performing ensemble verification

2.3.1.3.1 Add GenEnsProd to PROCESS_LIST

GenEnsProd will need to be added to the PROCESS_LIST in addition to EnsembleStat to generate the ensemble verification output.

```
PROCESS_LIST = ..., EnsembleStat, GenEnsProd, ...
```

2.3.1.3.2 Set input variables

Set the input dir and template variables for **GenEnsProd** to match the values set for FCST input to EnsembleStat. Also set the output dir to match EnsembleStat output dir.

```
GEN_ENS_PROD_INPUT_DIR = {FCST_ENSEMBLE_STAT_INPUT_DIR}
GEN_ENS_PROD_INPUT_TEMPLATE = {FCST_ENSEMBLE_STAT_INPUT_TEMPLATE}
GEN_ENS_PROD_OUTPUT_DIR = {ENSEMBLE_STAT_OUTPUT_DIR}
```

2.3.1.3.3 Set GenEnsProd output template to include filename

- **If the EnsembleStat output template is set**, then copy the value and add a template for the NetCDF output filename at the end following a forward slash '/' character.

  For example, if

  ```
  ENSEMBLE_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}
  ```

  then set

  ```
  GEN_ENS_PROD_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}/gen_ens_prod_{valid?fmt=%Y%m%d_%H%M
  ↪%S}V_ens.nc
  ```

  or something similar.

- **If the EnsembleStat output template is not set,** then set GenEnsProd's template to the desired NetCDF output filename. Here is an example:

  ```
  GEN_ENS_PROD_OUTPUT_TEMPLATE = gen_ens_prod_{valid?fmt=%Y%m%d_%H%M%S}V_ens.nc
  ```

Ensure that any downstream wrappers in the PROCESS_LIST are configured to read the correct GenEnsProd output file instead of the _ens.nc file that was previously generated by EnsembleStat.

2.3.1.3.4 Handle ENS_VAR variables

If ENS_VAR<n>_* variables are already set, this section can be skipped.

If ENS_VAR<n>_* variables are not set, add ENS_VAR<n> variables.

- If FCST_ENSEMBLE_STAT_VAR<n>_* variables are set, set the ENS_VAR<n>_* values to the same values.

  For example:

  ```
  ENS_VAR1_NAME = {FCST_ENSEMBLE_STAT_VAR1_NAME}
  ENS_VAR1_LEVELS = {FCST_ENSEMBLE_STAT_VAR1_LEVELS}
  ENS_VAR2_NAME = {FCST_ENSEMBLE_STAT_VAR2_NAME}
  ENS_VAR2_LEVELS = {FCST_ENSEMBLE_STAT_VAR2_LEVELS}
  ```

- If FCST_ENSEMBLE_STAT_VAR<n>_* variables are **not** set, set the ENS_VAR<n>_* values to the values set for the FCST_VAR<n>_*.

  For example:

  ```
  ENS_VAR1_NAME = {FCST_VAR1_NAME}
  ENS_VAR1_LEVELS = {FCST_VAR1_LEVELS}
  ENS_VAR2_NAME = {FCST_VAR2_NAME}
  ENS_VAR2_LEVELS = {FCST_VAR2_LEVELS}
  ```

2.3.1.3.5 Set GenEnsProd Variables

If any of the following ENSEMBLE_STAT_* variables are set in the configuration file, then rename them to the corresponding GEN_ENS_PROD_* variable. These are no longer valid settings for EnsembleStat.

Old Name	New Name
ENSEMBLE_STAT_NBRHD_PROB_WIDTH	GEN_ENS_PROD_NBRHD_PROB_WIDTH
ENSEMBLE_STAT_NBRHD_PROB_SHAPE	GEN_ENS_PROD_NBRHD_PROB_SHAPE
ENSEMBLE_STAT_NBRHD_PROB_VLD_THRESH	GEN_ENS_PROD_NBRHD_PROB_VLD_THRESH
ENSEMBLE_STAT_NMEP_SMOOTH_VLD_THRESH	GEN_ENS_PROD_NMEP_SMOOTH_VLD_THRESH
ENSEMBLE_STAT_NMEP_SMOOTH_SHAPE	GEN_ENS_PROD_NMEP_SMOOTH_SHAPE
ENSEMBLE_STAT_NMEP_SMOOTH_METHOD	GEN_ENS_PROD_NMEP_SMOOTH_METHOD
ENSEMBLE_STAT_NMEP_SMOOTH_WIDTH	GEN_ENS_PROD_NMEP_SMOOTH_WIDTH
ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_DX	GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_DX
ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_RADIUS	GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_RADIUS

If any of the following ENSEMBLE_STAT_* variables are set in the configuration file, then set the corresponding GEN_ENS_PROD_* variables to the same value or reference the ENSEMBLE_STAT_* version.

ENSEMBLE_STAT_N_MEMBERS
ENSEMBLE_STAT_ENS_THRESH
ENSEMBLE_STAT_REGRID_TO_GRID
ENSEMBLE_STAT_REGRID_METHOD
ENSEMBLE_STAT_REGRID_WIDTH
ENSEMBLE_STAT_REGRID_VLD_THRESH
ENSEMBLE_STAT_REGRID_SHAPE
FCST_ENSEMBLE_STAT_INPUT_GRID_DATATYPE

Example:

```
GEN_ENS_PROD_N_MEMBERS = {ENSEMBLE_STAT_N_MEMBERS}
GEN_ENS_PROD_ENS_THRESH = {ENSEMBLE_STAT_ENS_THRESH}
GEN_ENS_PROD_REGRID_TO_GRID = {ENSEMBLE_STAT_REGRID_TO_GRID}
GEN_ENS_PROD_REGRID_METHOD = {ENSEMBLE_STAT_REGRID_METHOD}
GEN_ENS_PROD_REGRID_WIDTH = {ENSEMBLE_STAT_REGRID_WIDTH}
GEN_ENS_PROD_REGRID_VLD_THRESH = {ENSEMBLE_STAT_REGRID_VLD_THRESH}
GEN_ENS_PROD_REGRID_SHAPE = {ENSEMBLE_STAT_REGRID_SHAPE}
GEN_ENS_PROD_INPUT_DATATYPE = {FCST_ENSEMBLE_STAT_INPUT_GRID_DATATYPE}
```

If any of the following ENSEMBLE_STAT_ENSEMBLE_FLAG_* variables are set in the configuration file, then rename them to the corresponding ENSEMBLE_STAT_NC_ORANK_FLAG_* variables AND add the corresponding GEN_ENS_PROD_ENSEMBLE_FLAG_* variables with the same value.

ENSEMBLE_STAT_ENSEMBLE_FLAG_LATLON
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN
ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT

For example, if

```
ENSEMBLE_STAT_ENSEMBLE_FLAG_LATLON = TRUE
```

then remove it and set

```
ENSEMBLE_STAT_NC_ORANK_FLAG_LATLON = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON = TRUE
```

Another example, if

```
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN = FALSE
```

then remove it and set

```
ENSEMBLE_STAT_NC_ORANK_FLAG_MEAN = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN = FALSE
```

2.3.1.3.6 Rename Variables

Old Name	New Name
ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV	GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV
ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS	GEN_ENS_PROD_ENSEMBLE_FLAG_MINUS
ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS	GEN_ENS_PROD_ENSEMBLE_FLAG_PLUS
ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN	GEN_ENS_PROD_ENSEMBLE_FLAG_MIN
ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX	GEN_ENS_PROD_ENSEMBLE_FLAG_MAX
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE	GEN_ENS_PROD_ENSEMBLE_FLAG_RANGE
ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY	GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY
ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP	GEN_ENS_PROD_ENSEMBLE_FLAG_NEP
ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP	GEN_ENS_PROD_ENSEMBLE_FLAG_NMEP

For further assistance, please navigate to the METplus Discussions page.

Chapter 3

Getting Started

This chapter reviews some important things to consider before starting a METplus project.

3.1 Questions to Consider

3.1.1 Questions to ask when applying METplus to a new testing and evaluation project

If a user is new to the concept of developing a verification or evaluations system, there are questions that should be considered to help determine which tools to use and how to set up METplus.

- First and foremost, what are the questions that need to be answered with this testing and evaluation project?
- What type of forecasts and type of observations will be used and how can/should they be matched?
  - What attributes of the forecast should be evaluated?
  - What is the standard for comparison that provides a reference level of skill (e.g., persistence, climatology, reference model)?
  - What is the geographic location of the model data being evaluated? Are there specific areas of interest for the evaluation?
  - What domain should be used to evaluate on: The model domain, observation domain (if gridded), or some other domain?
  - What is the evaluation time period? Retrospective with specific dates? Ongoing, near-real-time evaluation? Or both retrospective and real-time?
- How should the testing and evaluation project be broken down into METplus Use Cases? One large one or multiple smaller ones?
  - How will METplus be run? Manually? Scheduled, through cron? Automated via a workflow manager (e.g. Rocoto, EC-Flow, Rose-Cylc)?
  - Where will METplus be run? Local machine, project machine, HPC system, in the cloud (e.g. AWS)? Serial runs or parallelized?

This section will provide some guidance on how to use METplus based on the answers.

- What type of forecasts and type of observations will be used? Will they be gridded or point-based? The METplus tools that can be used will vary depending on the answer. Here's a matrix to help:

Table 1: METplus Tools Decision Matrix

	Gridded Forecast	Point Forecast
Gridded Observation/Analysis	Gen-Ens-Prod PCP-Combine Ensemble-Stat Grid-Stat Wavelet-Stat MODE MTD Series-Analysis Grid-Diag TC-Gen TC-RMW	Point-Stat (run with the Analyses as the "forecast" and Point Forecast as the "observation")
Point Observations	Point-Stat Ensemble-Stat	Stat-Analysis (run by passing in MPR records) TC-Pairs TC-Gen TC-Stat

- What attributes of the forecast should be evaluated?
  - This refers to defining the forecast fields to be evaluated, as well as the forecast characteristics such as bias, reliability, resolution, and prediction of events. It also means understanding the nature of the forecast and observations.

3.1.1.1 Examples of the nature of fields to be evaluated

- Continuous fields – the values change at the decimal level.
- Categorical fields – the values change incrementally most likely as integers or categories. Continuous fields can also be turned into categorical fields via applying thresholds.
- Probability fields – the values represent the probability or likelihood of an event occurring, usually represented by thresholds.
- Ensemble fields – are made up of multiple predictions either from the same modeling system or multiple systems.

Definitions of statistics categories associated with each type of field:

- Continuous statistics - measures how the values of the forecasts differ from the values of the observations.
  - METplus line types: SL1L2, SAL1L2, VL1L2, VAL1L2, CNT, VCNT.
  - METplus tools:
- Categorical statistics - measures how well the forecast captures events.
  - METplus line types: FHO, CTC, CTS, MCTC, MCTS, ECLV, TC stats, ExtraTC stats, TC-Gen stats.
- Probability statistics - measures attributes such as reliability, resolution, sharpness, and uncertainty.
  - METplus line types: PCT, PSTD, PJC, PRC.
- Ensemble statistics - measures attributes as the relationship between rank of observation and members, spread of ensemble member solutions and continuous measures of skill.

3.1.1.2 Additional verification and diagnostic approaches that can be helpful

- Geographical methods demonstrate where the error occurs geographically.
  - METplus methods: Series-Analysis tool.
  - METplus line types: Most Grid-Stat and Point-Stat line types.
- Object Based measures the location error of the forecast and how the total error break down into variety of descriptive attributes.
  - METplus methods: MODE, MTD, MvMODE, Grid-Stat Distance Maps.
  - METplus line types: MODE object attribute files, MODE CTS, MTD object attribute files, MTD CTS, Grid-Stat DMAP.
- Neighborhood relaxes the requirement for an exact match by evaluating forecasts in the local neighborhood of the observations.
  - METplus methods: Grid-Stat Neighborhood, Point-Stat HiRA, Ensemble-Stat HiRA.
  - METplus line types: NBRCTC, NBRCTS, NBRCNT, ECNT, ORANK, RPS.
- Domain Decomposition and Transform applies a transform to a given field to identify errors on different spatial scales:
  - METplus methods: Grid-Stat Fourier Decomposition; Wavelet-Stat tool, TC-RMW tool.
  - METplus line types: Grid-Stat SL1L2, SAL1L2, VL1L2, VAL1L2, CNT, VCNT; Wavelet Stat: ISC, RMW output file.
- Feature Relative identifies systematic errors associated with a group of case studies.
  - METplus methods: Feature Relative Use Cases.
- Relationship between two fields: generates a joint PDF between two fields.
  - METplus methods: Grid-Diag tool.
- Subseasonal-to-Seasonal Diagnostics compute indices to establish the ability of the model to predict S2S drivers.
  - METplus methods: S2S Use Cases.

3.1.1.3 What is the standard for comparison that provides a reference level of skill (e.g., persistence, climatology, reference model)?

Climatologies or Reference models may be passed into METplus using the following configuration options:

- {MET TOOL}_CLIMO_MEAN
- {MET TOOL}_CLIMO_STDEV
- {MET TOOL}_CLIMO_CDF

This can be found in Grid-Stat, Point-Stat, Gen-Ens-Prod, Series-Analysis, and Ensemble-Stat tools.

3.1.1.4 What is the geographic location of the model data being evaluated? Are there specific areas of interest for the evaluation?

Masking regions are what METplus uses to define verification areas of interest. These can be defined prior to running tools using the Gen-Vx-Mask tool, or during run-time using the METPLUS_MASK_DICT options.

3.1.1.5 What domain should be used for evaluation: The model domain, observation domain (if gridded), or some other domain?

The decision to evaluate on model or observation/analysis domain is user-specific but the user may want to consider the following:

- Regridding to the courser domain will smooth high resolution information that may be important but smoother forecasts tend to score better.
- Regridding to a finer domain essentially adds in additional information that is not real.
- One way to avoid the interpolation debate is to regrid both to a third grid.

Regridding in METplus can be completed using the Regrid-Data-Plane tool if the fields will be used more than once.

Regridding can also be done on the fly using the {Tool}_REGRID_TO_GRID. All grid-to-grid verification tools have the regridding capability in it.

3.1.1.6 What is the evaluation time period? Retrospective with specific dates? Ongoing, near-real-time evaluation? Or both retrospective and realtime?

Basically, running retrospectively means that the observations/analyses are already available on disk and running in realtime is when the system needs to wait for the observations to be available on the system.

In METplus, the LOOP_BY configuration can be used.

LOOP_BY = VALID or REALTIME to have METplus proceed through the data based on Valid Time.

LOOP_BY = INIT or RETRO to have METplus proceed through the data based on Initialization Time.

3.1.1.7 How should the testing and evaluation project be broken down into METplus Use Cases? One large one or multiple smaller ones?

A single use case is typically appropriate for a given evaluation so that all of the information is found in one configuration file. However, users may want to evaluate different combinations of models and observations. For example, they may want to compare forecastA with observationA, forecastA with observationB, forecastB with observationA, forecastB with observationB, etc. In this case, separate METplus configuration files can be created with information specific to each forecast or observation. Another configuration file can be used to control settings common to each evaluation, such as timing information and the process list. The METplus wrappers can be called with each desired combination.

```
run_metplus.py forecastA.conf observationA.conf use_case_name.conf
run_metplus.py forecastA.conf observationB.conf use_case_name.conf
run_metplus.py forecastB.conf observationA.conf use_case_name.conf
run_metplus.py forecastB.conf observationB.conf use_case_name.conf
```

It is also worth considering the Use Case Rules. A case may be affected by the size of the data, the length of time to run and other factors.

3.1.1.8 How will METplus be run? Manually? Scheduled through cron? Automated via a workflow manger (e.g. Rocoto, EC-Flow, Rose-Cylc)?

- If run manually, this can be done.
- If scheduled through cron, a bash or csh script can be written to set up environment variables to pass into METplus.
- If automated via a workflow manager, it is recommended the user consider configuring the use cases to run smaller amounts of data.

3.1.1.9 Where will METplus be run? Local machine, project machine, HPC system, in the cloud (e.g. AWS)? Serial runs or parallelized?

- Running on linux or a project machine – identify where METplus is installed by running **which run_metplus.py**; it is recommended an additional user.conf or system.conf file is passed into the **run_metplus.py** to direct where output should be written.
- Running on HPC systems - check with the system admin to see if it has been configured as a module and how to load netCDF and Python modules. For NOAA and NCAR HPCs systems, please refer to the Existing Builds pages for the desired version for instructions on how to load the METplus related modules.
- Running on Cloud (AWS) - these instructions are coming soon.
- Running in parallel - As of MET v10.1.0 Grid-Stat can be run in parallel. Please reach out via METplus Discussions if help is needed.

3.1.1.10 Would a flowchart help to provide clarity?

Utilizing a flowchart can assist in identifying which verification steps can be completed by which METplus tools.

3.2 Running METplus

3.2.1 Example Wrapper Use Case

- Create a *User Configuration File* (page 45) (named user_system.conf in this example).
- Run the Example Wrapper use case. In a terminal, run:

```
run_metplus.py \
/path/to/METplus/parm/use_cases/met_tool_wrapper/Example/Example.conf \
/path/to/user_system.conf
```

Replacing */path/to/user_system.conf* with the path to the user configuration file and */path/to/METplus* with the path to the location where METplus is installed.

The last line of the screen output should match this format:

```
05/04 09:42:52.277 metplus INFO: METplus has successfully finished running.
```

If this log message is not shown, there is likely an issue with one or more of the default configuration variable overrides in the *User Configuration File* (page 45).

This use case does not utilize any of the MET tools, but simply demonstrates how the *Common Config Variables* (page 46) control a use case run.

If the run was successful, the line above the success message should contain the path to the METplus log file that was generated:

```
05/04 09:44:21.534 metplus INFO: Check the log file for more information: /path/to/output/
↪logs/metplus.log.20210504094421
```

- Review the log file and compare it to the Example.conf use case configuration file to see how the settings correspond to the result.
- Review the *metplus_final.conf* (page 40) file to see all of the settings that were used in the use case.

3.2.2 GridStat Wrapper Basic Use Case

- *Obtain sample input data* (page 33) for the **met_tool_wrapper** use cases. The tarfile should be in the directory that corresponds to the major/minor release and starts with sample_data-met_tool_wrapper.
- Create a *User Configuration File* (page 45) (named user_system.conf in this example). Ensure that **INPUT_BASE** is set to the directory where the sample data tarfile was uncompressed.
- Run the GridStat Wrapper basic use case. In a terminal, run:

```
run_metplus.py \
/path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat.conf \
/path/to/user_system.conf
```

Replacing */path/to/user_system.conf* with the path to the user configuration file and */path/to/METplus* with the path to the location where METplus is installed.

If the run was successful, the line above the success message should contain the path to the METplus log file that was generated.

- Review the log file and compare it to the **GridStat.conf** use case configuration file to see how the settings correspond to the result.
- Review the *metplus_final.conf* (page 40) file to see all of the settings that were used in the use case.

3.3 METplus in Docker

METplus is available on DockerHub. The METplus Docker image includes all of the MET executables from the corresponding METplus Coordinated Release and the METplus wrappers are pre-configured to use them.

To pull the latest official release, run:

```
docker pull dtcenter/metplus:latest
```

Tags for previous releases and development releases are also available. Refer to the list of available tags on DockerHub.

3.3.1 Sample Input Data

Sample input data for all of the use cases provided with the METplus wrappers are also available on DockerHub. These data are found in the dtcenter/metplus-data DockerHub repository and are named with the X.Y version of the corresponding METplus Coordinated Release and the name of the use case category separated by a dash, e.g. 4.1-data_assimilation or 4.0-met_tool_wrapper. A list of available tags for input data can also be found on DockerHub.

To make these data available in a METplus Docker container, first create a Docker data volume from the desired tag and give it a name with the *–name* argument:

```
docker create --name met_tool_wrapper dtcenter/metplus-data:4.1-met_tool_wrapper
```

Then mount the data volume to the container using the *–volumes-from* argument to the *docker run* command:

```
docker run --rm -it --volumes-from met_tool_wrapper dtcenter/metplus:4.1.4 bash
```

The input data will be available inside the container under /data/input/METplus_Data.

3.3.2 Running METplus in Docker

The run_metplus.py script is in the user's path inside the container. The use case configuration files can be found under /metplus/METplus/parm/use_cases. The values for *MET_INSTALL_DIR* (page 38), *INPUT_BASE* (page 39), and *OUTPUT_BASE* (page 39) are set to appropriate default values. A use case can be run by simply passing the use case configuration file to the run script:

```
run_metplus.py /metplus/METplus/parm/use_cases/met_tool_wrapper/Example/Example.conf
```

Output from the use case will be written to /data/output by default. The value for *OUTPUT_BASE* (page 39) can be changed to write elsewhere:

```
run_metplus.py /metplus/METplus/parm/use_cases/met_tool_wrapper/Example/Example.conf config.
↪OUTPUT_BASE=/data/output/my_data_dir
```

If *Sample Input Data* (page 27) is mounted to the container, then use cases from the corresponding category can be run:

```
run_metplus.py /metplus/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat.conf
```

Please note that use cases that have additional Python package dependencies may not run successfully unless those packages are installed inside the container or obtained elsewhere.

Chapter 4

Installation

4.1 Introduction

This chapter describes how to download and set up METplus Wrappers.

4.2 Supported architectures

METplus Wrappers was developed on Debian Linux and is supported on this platform. Each release listed on the METplus Downloads page includes a link to the **Existing Builds and Docker** for that version. The METplus team supports the installation of the METplus components on several operational and research high performance computing platforms, including those at NCAR, NOAA, and other community machines. Pre-built METplus images on DockerHub are also provided.

4.3 Programming/scripting languages

METplus Wrappers is written in Python. It is intended to be a tool for the modeling community to use and adapt. As users make upgrades and improvements to the tools, they are encouraged to offer those upgrades to the broader community by offering feedback to the developers or coordinating for a GitHub pull. For more information on contributing code to METplus Wrappers, please create a post in the METplus GitHub Discussions Forum.

4.4 Requirements

4.4.1 Software Requirements

Minimum Requirements

The following software is required to run METplus Wrappers:

- Python 3.8.6 or above

- MET version 11.0.0 or above - For information on installing MET please see the Software Installation/Getting Started section of the MET User's Guide.

Wrapper Specific Requirements

- TCMPRPlotter wrapper
  - R version 3.2.5
- SeriesAnalysis wrapper
  - convert (ImageMagick) utility - if generating plots and/or animated images from the output
- PlotDataPlane wrapper
  - convert (ImageMagick) utility - if generating images from the Postscript output

4.4.2 Python Package Requirements

The version number listed next to any Python package corresponds to the version that was used for testing purposes. Other versions of the packages **may** still work but it is not guaranteed. Please install these packages using pip or conda.

Minimum Requirements

To run most of the METplus wrappers, the following packages are required:

- dateutil (2.8)

Using pip:

```
pip3 install python-dateutil==2.8
```

Using conda:

```
conda install -c conda-forge python-dateutil=2.8
```

MET Python Embedding Requirements

If running use cases that use Python embedding, the **MET** executables must be installed with Python enabled.

See Appendix F Python Embedding section in the MET User's Guide for more information.

Wrapper Specific Requirements

The following wrappers require that additional Python packages be installed to run.

- SeriesAnalysis wrapper
  - netCDF4 (1.5.4)
- CyclonePlotter wrapper
  - cartopy (0.20.3)
  - matplotlib (3.5.2)

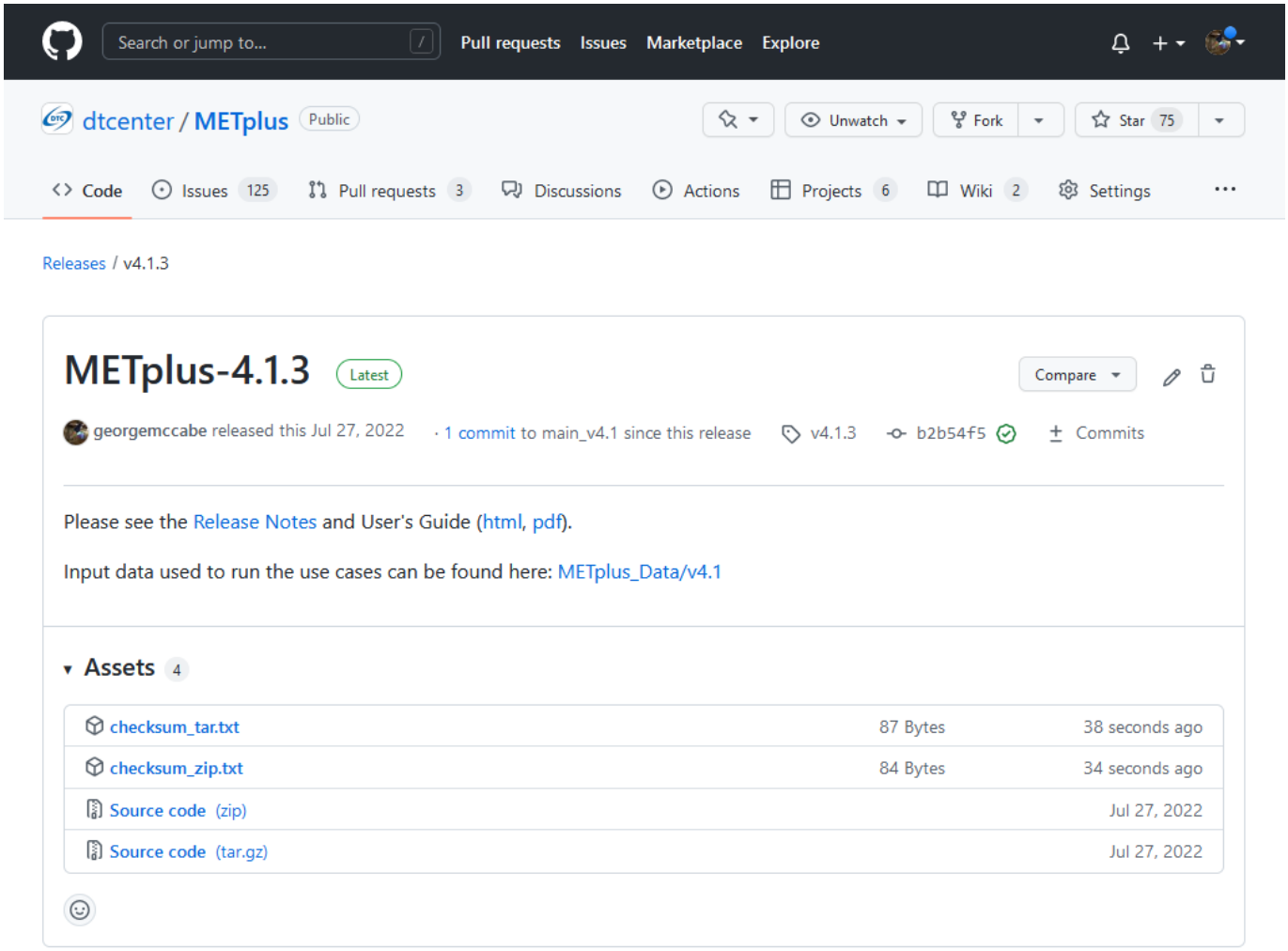
Cartopy, one of the dependencies of CyclonePlotter, attempts to download shapefiles from the internet to complete successfully. So if CyclonePlotter is run on a closed system (i.e. no internet), additional steps need to be taken. First, go to the Natural Earth Data webpage and download the small scale (1:110m) cultural and physical files that will have multiple extensions (e.g. .dbf, .shp, .shx). Untar these files in a noted location. Finally, create an environment variable in the user-specific system configuration file for CARTOPY_DIR, setting it to the location where the shapefiles are located.

4.5 Getting the METplus Wrappers source code

The METplus Wrappers source code is available for download from the public GitHub repository. The source code can be retrieved either through a web browser or the command line.

4.5.1 Get the source code via Web Browser

- Create a directory where the METplus Wrappers will be installed
- Open a web browser and go to the latest stable METplus release.



- Click on the 'Source code' link (either the *zip* or *tar.gz*) under Assets and when prompted, save it to the directory.
- (Optional) Verify the checksum of the source code download
  - Download the checksum file that corresponds to the source code download link that was used (checksum_zip.txt for the *zip* file and checksum_tar.txt for the *tar.gz* file). Put the checksum file into the same directory as the source code file.
  - Run the *sha256sum* command with the –check argument to verify that the source code download file was not corrupted.

Zip File:

```
sha256sum --check checksum_zip.txt
```

Tar File:

```
sha256sum --check checksum_tar.txt
```

Note: If the source code is downloaded using **wget**, then the filenames will not match the filenames listed in the checksum files. If the source code is downloaded using **curl**, the *-LJO* flags should be added to the command to preserve the expected filenames found in the checksum files.

- Uncompress the source code (on Linux/Unix*: gunzip* for zip file or *tar xvfz* for the tar.gz file)

4.5.2 Get the source code via Command Line

- Open a shell terminal
- Clone the DTCenter/METplus GitHub repository:

SSH:

```
git clone git@github.com:dtcenter/metplus
```

HTTPS:

```
git clone https://github.com/dtcenter/metplus
```

4.6 Obtain sample input data

The use cases provided with the METplus release have sample input data associated with them. This step is optional but is required to be able to run the example use cases, which illustrate how the wrappers work.

- Create a directory to put the sample input data. This will be the directory to set for the value of INPUT_BASE in the METplus Configuration.
- Go to the web page with the sample input data.
- Click on the vX.Y version directory that corresponds to the release to install, i.e. v4.0 directory for the v4.0.0 release.
- Click on the sample data tgz file for the desired use case category or categories run and when prompted, save the file to the directory created above.

Note: Files with the version number in the name, i.e. sample_data-data_assimilation-4.0.tgz, have been updated since the last major release. Files without the version number in the file name have not changed since the last major release and can be skipped if the data have already been obtained with a previous release.

4.7 METplus Wrappers directory structure

The METplus Wrappers source code contains the following directory structure:

```
METplus/
  build_components/
  docs/
  internal/
  manage_exernals/
  metplus/
  parm/
  produtil/
  ush/
```

The top-level METplus Wrappers directory consists of a README.md file and several subdirectories.

The **build_components/** directory contains scripts that use manage_externals and files available on dtcenter.org to download MET and start the build process.

The **docs/** directory contains documentation for users and contributors (HTML) and Doxygen files that are used to create the METplus wrapper API documentation. The Doxygen documentation can be created and viewed via web browser if the developer has Doxygen installed on the host. The Doxygen documentation is useful to contributors and is not necessary for METplus end-users.

The **internal/** directory contains scripts that are only relevant to METplus developers and contributors, such as tests and files used with Docker.

The **manage_externals/** directory contains scripts used to facilitate the downloading and management of components that METplus interacts with such as MET and METviewer.

The **metplus/** directory contains the wrapper scripts and utilities.

The **parm/** directory contains all the configuration files for MET and METplus Wrappers.

The **produtil/** directory contains part of the external utility produtil.

The **ush/** directory contains the run_metplus.py script that is executed to run use cases.

4.8 External Components

4.8.1 GFDL Tracker (optional)

- The standalone Geophysical Fluid Dynamics Laboratory (GFDL) vortex tracker is a program that objectively analyzes forecast data to provide an estimate of the vortex center position (latitude and longitude), and track the storm for the duration of the forecast.
- Visit https://dtcenter.org/community-code/gfdl-vortex-tracker for more information
  - See the manage externals section of this documentation to download the GFDL vortex tracker automatically as part of the system.
  - To download and install the tracker locally, get http://dtcenter.org/sites/default/files/community-code/gfdl/standalone_gfdl-vortextracker_v3.9a.tar.gz and follow the instructions listed in that archive to build on a local system.
  - Instructions on how to configure and use the GFDL tracker are found here https://dtcenter.org/sites/default/files/community-code/gfdl/standalone_tracker_UG_v3.9a.pdf

4.9 Disable UserScript wrapper (optional)

The UserScript wrapper allows any shell command or script to be run as part of a METplus use case. It is used to preprocess/postprocess data or to run intermediate commands between other wrappers.

If desired, this wrapper can be disabled upon installation to prevent security risks. To disable the UserScript wrapper, simply remove the following file from the installation location:

```
METplus/metplus/wrapper/user_script_wrapper.py
```

Please note that use cases provided with the METplus repository that utilize the UserScript wrapper will fail if attempted to run after it has been disabled.

4.10 Add ush directory to shell path (optional)

To call the run_metplus.py script from any directory, add the ush directory to the path. The following commands can be run in a terminal. They can also be added to the shell run commands file (.cshrc for csh/tcsh or .bashrc for bash). For the following commands, change **/path/to** to the actual path to the METplus directory on the local file system.

csh/tcsh:

```
# Add METplus to path
set path = (/path/to/METplus/ush $path)
```

bash/ksh:

```
# Add METplus to path
export PATH=/path/to/METplus/ush:$PATH
```

4.11 Set Default Configuration File for Shared Install

The default METplus configurations are found in *parm/metplus_config/defaults.conf*. If configuring METplus Wrappers in a common location for multiple users, it is recommended that the values for **MET_INSTALL_DIR** and **INPUT_BASE** are set in the default configuration file. More information on how to set these values can be found in the *Default Configuration File section* (page 38) in the next chapter.

Chapter 5

Configuration

This chapter is a guide on configuring METplus Wrappers.

5.1 Config Best Practices / Recommendations

- Set the log level (*LOG_LEVEL* (page 43)) to an appropriate level. Setting the value to DEBUG will generate more information in the log output. Users are encouraged to run with DEBUG when getting started with METplus or when investigating unexpected behavior.
- Set *SCRUB_STAGING_DIR* to False to preserve intermediate files to help with debugging issues.
- Review the log files to verify that all of the processes ran cleanly. Some log output will be written to the screen, but the log files contain more information, such as log output from the MET tools.
- The order in which METplus config files are read by run_metplus.py matters. Each subsequent config file defined on the command line will override any values defined in an earlier config file. It is recommended to create a *User Configuration File* (page 45) and pass it to the script last to guarantee that those values are used in case any variables are accidentally defined in multiple conf files.
- Check the metplus_final.conf (see *METPLUS_CONF* (page 40)) file to verify that all variables are set to the expected value, as it contains all the key-values that were specified.
- If configuring METplus Wrappers in a common location for multiple users:
  - It is recommended that the values for **MET_INSTALL_DIR** and **INPUT_BASE** are changed to valid values in the *Default Configuration File* (page 38).
  - It is recommended to leave **OUTPUT_BASE** set to the default value in the *Default Configuration File* (page 38). This prevents multiple users from accidentally writing to the same output directory.
- If obtaining the METplus Wrappers with the intention of updating the same local directory as new versions become available, it is recommended to leave all default values in the *Default Configuration File* (page 38) unchanged and set them in a *User Configuration File* (page 45) that is passed into every call to run_metplus.py. This is done to avoid the need to change the default values after every update.

5.2 Default Configuration File

The default METplus configurations are found in *parm/metplus_config/defaults.conf*. These settings are automatically loaded at the start of a METplus Wrappers run and do not need to be invoked on the command line.

These settings include:

- Location of MET installation
- Directories where input data are located
- Directory to write output data and temporary files
- Logging levels for METplus wrapper and MET application output
- Location of other non-MET executables/binaries

The values in this file can either be set directly in this file or in a *User Configuration File* (page 45).

5.2.1 Required (/path/to)

Some of the variables in this file must be changed from the default value before running. These variables are set to **/path/to** by default and are described below. Running METplus with **/path/to** configuration entries present results in an error.

5.2.1.1 MET_INSTALL_DIR

The MET installation directory is the location where the MET tools are installed on the system. This directory is typically named 'met' or 'met-X.Y' or 'met-X.Y.Z' and should contain at least two directories: **share** and **bin** (or **exec** on some installations). The **bin** directory will contain the MET executables, such as grid_stat.

```
>>>ls /usr/local/met
bin  share
>>>
>>>ls /usr/local/met/bin
ascii2nc       grid_diag      mode           plot_data_plane    rmw_analysis      tc_pairs
ensemble_stat  grid_stat      mode_analysis  plot_mode_field    series_analysis   tc_rmw
gen_vx_mask    gsid2mpr       modis_regrid   plot_point_obs     shift_data_plane  tc_stat
gis_dump_dbf   gsidens2orank  mtd            point2grid         stat_analysis     wavelet_stat
gis_dump_shp   lidar2nc       pb2nc          point_stat         tc_dland          wwmca_plot
gis_dump_shx   madis2nc       pcp_combine    regrid_data_plane  tc_gen            wwmca_regrid
>>>
>>>ls /usr/local/met/share/met
colortables  config  map  poly  ps  python  Rscripts  table_files  tc_data  version.txt  wrappers
```

Based on the directory listing output above, the following should be set:

```
MET_INSTALL_DIR = /usr/local/met
```

For information on installing MET please see the Software Installation/Getting Started section of the MET User's Guide.

5.2.1.2 INPUT_BASE

The input base is the directory that contains the sample input data used to run the use case examples found in the parm/use_cases directory. This directory should contain one or more of the following:

- A directory called **model_applications** which contains directories that correspond to each use case directory under *parm/use_cases/model_applications*
- A directory called **met_test** which contains data used for the use cases found under *parm/use_cases/met_tool_wrapper*

```
>>>ls /d1/METplus_Data
met_test  model_applications
```

Based on the directory listing output above, the following should be set:

```
INPUT_BASE = /d1/METplus_Data
```

5.2.1.3 OUTPUT_BASE

The output base is the directory where logs and output files are written. This should be set to a path where the user running the METplus wrappers has permission to write files. The directory will be created automatically if it does not exist already.

Example:

```
OUTPUT_BASE = /d1/user/output
```

5.2.2 Optional

5.2.2.1 MET_BIN_DIR

The MET bin directory contains all of the MET executables, like grid_stat. Typically this is a directory under *MET_INSTALL_DIR* (page 38) named **bin**. This is the default value:

```
MET_BIN_DIR = {MET_INSTALL_DIR}/bin
```

However, some environments require these files to be contained in a directory named **exec** instead. If this is the case for the MET installation, then change the value appropriately:

```
MET_BIN_DIR = {MET_INSTALL_DIR}/exec
```

5.2.2.2 METPLOTPY_BASE (user_env_vars)

This is the path to the location where METplotpy is installed. The variable is found under the [user_env_vars] section heading, which will set it as an environment variable. See *User Environment Variables* (page 89) for more information on the [user_env_vars] section. This variable is referenced in some METplotpy functions. It is not necessary to set this variable if METplotpy will not be used or if it is already set in the user's environment.

5.2.2.3 METPLUS_CONF

This is the path to the final METplus configuration file that contains the full list of all configuration variables set for a given run. This includes all of the values set by the METplus configuration files that were passed into the script, as well as the values from the *Default Configuration File* (page 38) and any default values set by the wrappers. This file is useful to review for debugging to see which values were actually used for the run. If a value set in the final conf differs from what was set in a configuration file passed to run_metplus.py, there is a good chance that this variable is set in another configuration file that was passed in afterwards.

The default value is a file called metplus_final.conf followed by the log timestamp (see *LOG_TIMESTAMP*) that is written in the *OUTPUT_BASE* (page 39) directory:

```
METPLUS_CONF = {OUTPUT_BASE}/metplus_final.conf.{LOG_TIMESTAMP}
```

This value is rarely changed, but it can be if desired.

5.2.2.4 TMP_DIR

Directory to write any temporary files created by the MET applications. By default, this is a directory inside the *OUTPUT_BASE* (page 39) directory:

```
TMP_DIR = {OUTPUT_BASE}/tmp
```

This value is rarely changed, but it can be if desired.

5.2.2.5 STAGING_DIR

Directory to write files that have been uncompressed or converted by the wrapper scripts. Files are written to this directory to prevent corrupting input data directories in case something goes wrong. File list ASCII files that contain a list of file paths to pass into MET tools such as MODE-TimeDomain or SeriesAnalysis are also written to this directory. See *FILE_LISTS_DIR* for more information.

By default this is a directory called **stage** inside the *OUTPUT_BASE* (page 39) directory:

```
STAGING_DIR = {OUTPUT_BASE}/stage
```

This value is rarely changed, but it can be if desired.

5.2.2.6 SCRUB_STAGING_DIR

True or False variable to determine if the *STAGING_DIR* should be removed after the METplus has finished running.

5.2.2.7 OMP_NUM_THREADS

If the MET executables were installed with threading support, then the number of threads used by the tools can be configured with this variable. See the glossary entry for *OMP_NUM_THREADS* for more information.

5.2.2.8 CONVERT

Location of the ImageMagick utility called **convert** used by PlotDataPlane and SeriesAnalysis wrappers to generate images from Postscript files. The default value is the name of the executable:

```
CONVERT = convert
```

If the executable is in the user's path, then this value does not need to be changed. However, if the tool is not in the user's path but is still available on the file system, this value can be set to the full path of the file.

5.2.2.9 GEMPAKTOCF_JAR

Path to the GempakToCF.jar file used to convert GEMPAK data to NetCDF format. This is only used if running a use case that reads GEMPAK data. The value should be set to the full path of the JAR file. The file can be found here: https://dtcenter.org/sites/default/files/community-code/metplus/utilities/GempakToCF.jar

5.2.3 Logging

5.2.3.1 Log File Information

Where to write logs files

5.2.3.1.1 LOG_METPLUS

This defines the name of the METplus log file:

```
LOG_METPLUS = {LOG_DIR}/metplus.log.{LOG_TIMESTAMP_TEMPLATE}
```

The value references *LOG_DIR* (page 42) and *LOG_TIMESTAMP_TEMPLATE* (page 42).

5.2.3.1.2 LOG_DIR

This defines the directory that will contain log files. Typically this is set to a directory called "logs" inside the *OUTPUT_BASE* directory:

```
LOG_DIR = {OUTPUT_BASE}/logs
```

The value can be changed if another location to write log files is preferred.

5.2.3.1.3 LOG_TIMESTAMP_TEMPLATE

Sets the desired timestamp format, using strftime format directives. It must only contain valid strftime format directives (see https://strftime.org). The current run time is substituted using the format specified unless *LOG_TIMESTAMP_USE_DATATIME* (page 42) is set to true/yes. By default, a new log file is created for each METplus run:

```
LOG_TIMESTAMP_TEMPLATE = %Y%m%d%H%M%S
```

This example will use the format YYYYMMDDHHMMSS, i.e. 20141231101159. Change this value to adjust the frequency that new log files are created. For example, to write all log output that is generated within a day to a single log file, set:

```
LOG_TIMESTAMP_TEMPLATE = %Y%m%d
```

This example will use the format YYYYMMDD, i.e. 20141231

5.2.3.1.4 LOG_TIMESTAMP_USE_DATATIME

If set to false/no (default), write log timestamps using the current time when the METplus run was started:

```
LOG_TIMESTAMP_USE_DATATIME = no
```

If set to true/yes, write log timstamps using the value set for *INIT_BEG* or *VALID_BEG* depending on the value set for *LOOP_BY*. This is useful if it is desired to organize the log output files based on the data that was processed during the run.

5.2.3.1.5 LOG_MET_OUTPUT_TO_METPLUS

If set to true/yes (default), log output from MET applications are written to the METplus log file:

```
LOG_MET_OUTPUT_TO_METPLUS = yes
```

If set to false/no, the output is written to a separate file in the log directory named after the application.

5.2.3.2 Log Level Information

How much information to log

5.2.3.2.1 LOG_LEVEL

This controls the level of logging output from the METplus wrappers that is written to the log file defined by *LOG_METPLUS* (page 41). It does not control the logging level of the actual MET applications. The possible values to:

- CRITICAL
- ERROR
- WARNING
- INFO
- DEBUG
- NOTSET

The default logging level is INFO:

```
LOG_LEVEL = INFO
```

The log output will contain messages from the level selected and above. If a use case is producing errors, then setting:

```
LOG_LEVEL = DEBUG
```

will produce additional logging output that is helpful to discover the cause of the error.

5.2.3.2.2 LOG_LEVEL_TERMINAL

This controls the level of logging that is output to the screen. The valid values are the same as *LOG_LEVEL* (page 43).

5.2.3.2.3 LOG_MET_VERBOSITY

This controls the logging verbosity level for all of the MET applications. The value can be set to an integer. Higher values produce more log output. The logging verbosity can also be set individually for each MET tool if more log output is desired for a specific application. For example:

```
LOG_MET_VERBOSITY = 2
LOG_ASCII2NC_VERBOSITY = 3
LOG_POINT_STAT_VERBOSITY = 4
```

In the above example, ASCII2NC will use 3, PointStat will use 4, and all other MET applications with use 2.

5.2.3.3 Log Formatting Information

How to format lines in log files

Note: The following variables control the format of the METplus log output that is written to the log files. It does not control the format of the log output that is written to the screen as standard output.

For more information on acceptable values, see the Python documentation for LogRecord: https://docs.python.org/3/library/logging.html#logging.LogRecord

5.2.3.3.1 LOG_INFO_LINE_FORMAT

This defines the format of the INFO log messages. Setting the value to:

```
LOG_INFO_LINE_FORMAT = %(asctime)s.%(msecs)03d %(name)s %(levelname)s: %(message)s
```

Produces a log file with INFO lines that match this format:

```
04/29 15:54:22.413 metplus INFO: Completed METplus configuration setup.
```

The format of the timestamp is set by *LOG_LINE_DATE_FORMAT* (page 45).

5.2.3.3.2 LOG_ERR_LINE_FORMAT

This defines the format of the ERROR log messages. Setting the value to:

```
LOG_ERR_LINE_FORMAT = %(asctime)s.%(msecs)03d %(name)s (%(filename)s:%(lineno)d)
↪%(levelname)s: %(message)s
```

Produces a log file with ERROR lines that match this format:

```
04/29 16:03:34.858 metplus (run_util.py:192) ERROR: METplus has finished running but had 1␣
↪error.
```

The format of the timestamp is set by *LOG_LINE_DATE_FORMAT* (page 45).

5.2.3.3.3 LOG_DEBUG_LINE_FORMAT

This defines the format of the DEBUG log messages. Setting the value to:

```
LOG_DEBUG_LINE_FORMAT = %(asctime)s.%(msecs)03d %(name)s (%(filename)s:%(lineno)d)
↪%(levelname)s: %(message)s
```

Produces a log file with DEBUG lines that match this format:

```
04/29 15:54:22.851 metplus (run_util.py:177) DEBUG: METplus took 0:00:00.850983 to run.
```

The format of the timestamp is set by *LOG_LINE_DATE_FORMAT* (page 45).

5.2.3.3.4 LOG_LINE_DATE_FORMAT

This defines the format of the timestamps used in the METplus log messages.

Setting the value to:

```
LOG_LINE_DATE_FORMAT = %m/%d %H:%M:%S
```

Produces a log file with timestamps that match this format:

```
04/29 15:54:22.851
```

5.2.3.3.5 LOG_LINE_FORMAT

Defines the default formatting of each METplus log output line. By default, this variable is referenced in *LOG_ERR_LINE_FORMAT* (page 44) and *LOG_DEBUG_LINE_FORMAT* (page 44).

5.3 User Configuration File

It is recommended that users create a METplus configuration file for each system that they are running the METplus wrappers. The file can be passed into run_metplus.py after any *use case configuration files* (page 46) so that the settings are applied to every use case that is run. Multiple user configuration files can also be created on a system to customize different work environments. At a minimum, a user configuration file should set the *OUTPUT_BASE* (page 39) variable so that output files are created in a familiar directory.

A minimal user configuration file contains:

```
[config]
OUTPUT_BASE = /my/output/base
```

where /my/output/base is a path where the user has write permission.

If using an installation of the METplus wrappers that does not have *MET_INSTALL_DIR* (page 38) and/or *INPUT_BASE* (page 39) set in the *default configuration file* (page 38), or if a different value for either variable is desired, it is appropriate to override these variables in a user configuration file:

```
[config]
OUTPUT_BASE = /my/output/base
INPUT_BASE = /my/input/base
MET_INSTALL_DIR = /usr/local/met-10.0.0
```

Overriding MET_INSTALL_DIR in the user configuration file allows users to use a older version or test a new beta version of MET. Overriding INPUT_BASE can be useful when developing a new use case.

Any other METplus configuration variables that are intended to be set for each run can be added to this file to the user's taste. *Logging* (page 41) configuration variables are often set in these files, most commonly *LOG_LEVEL* (page 43) = DEBUG to produce additional log output.

5.4 Use Case Configuration Files

Example configuration files that contain settings to run various use cases can be found in the *parm/use_cases* directory. There are two directories inside this directory:

- **met_tool_wrapper** contains simple use cases that run one wrapper at a time. They provide examples of how to configure and run a single wrapper to help users become familiar with the configurations that are available for that wrapper.
- **model_applications** contains directories organized by category. These use cases often run multiple wrappers in succession to demonstrate how the tools can be used in more complex verification workflows by end users.

The use case configuration files found in these directories contain *Common Config Variables* (page 46) that define each use case. Configuration variables that are specific to a user's environment (INPUT_BASE, OUTPUT_BASE, MET_INSTALL_DIR, etc.) are not *set* in these files. However, INPUT_BASE and OUTPUT_BASE are *referenced* by variables that are found in these files. For example:

```
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
...
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/GridStat
```

All input data read by the use case is relative to INPUT_BASE and all output paths for data written by the use case is relative to OUTPUT_BASE. The expectation is a use case can be run locally if the user's INPUT_BASE contains the sample data associated with the use case *AND* any additional dependencies (i.e. Python packages) are available. See the chapter titled *METplus Use Cases* (page 309) to view the documentation for the existing use cases to see if additional dependencies are required for a given use case.

More information about the variables set in the use case configuration files can be found in the *Common Config Variables* (page 46) section.

5.5 Common Config Variables

5.5.1 Timing Control

This section describes the METplus wrapper configuration variables that are used to control which times are processed. It also covers functionality that is useful for processing data in realtime by setting run times based on the clock time when the METplus wrappers are run.

5.5.1.1 LOOP_BY

The METplus wrappers can be configured to loop over a set of valid times or a set of initialization times. This is controlled by the configuration variable called *LOOP_BY*. If the value of this variable is set to INIT or RETRO, looping will be relative to initialization time. If the value is set to VALID or REALTIME, looping will be relative to valid time.

5.5.1.2 Looping by Valid Time

When looping over valid time (*LOOP_BY* = VALID or REALTIME), the following variables must be set:

5.5.1.2.1 VALID_TIME_FMT

This is the format of the valid times the user can configure in the METplus Wrappers. The value of *VALID_BEG* and *VALID_END* must correspond to this format.

Example:

```
VALID_TIME_FMT = %Y%m%d%H
```

Using this format, the valid time range values specified must be defined as YYYYMMDDHH, i.e. 2019020112.

5.5.1.2.2 VALID_BEG

This is the first valid time that will be processed. The format of this variable is controlled by *VALID_TIME_FMT*. For example, if VALID_TIME_FMT=%Y%m%d, then VALID_BEG must be set to a valid time matching YYYYMMDD, such as 20190201.

5.5.1.2.3 VALID_END

This is the last valid time that can be processed. The format of this variable is controlled by *VALID_TIME_FMT*. For example, if VALID_TIME_FMT=%Y%m%d, then VALID_END must be set to a valid time matching YYYYMMDD, such as 20190202.

Note: The time specified for this variable will not necessarily be processed. It is used to determine the cutoff of run times that can be processed. For example, if METplus Wrappers is configured to start at 20190201 and end at 20190202 processing data in 48 hour increments, it will process valid time 20190201 then increment the run time to 20190203. This is later than the VALID_END value, so execution will stop. However, if the increment is set to 24 hours (see *VALID_INCREMENT*), then METplus Wrappers will process valid times 20190201 and 20190202 before ending execution.

5.5.1.2.4 VALID_INCREMENT

This is the time interval to add to each run time to determine the next run time to process. See *Time Interval Units* (page 52) for information on time interval formatting. Units of hours are assumed if no units are specified. This value must be greater than or equal to 60 seconds because the METplus wrappers currently do not support processing intervals of less than one minute.

The following is a configuration that will process valid time 2019-02-01 at 00Z until 2019-02-02 at 00Z in 6 hour (21600 seconds) increments:

```
[config]
LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2019020100
VALID_END = 2019020200
VALID_INCREMENT = 6H
```

Note: Substituting VALID_INCREMENT = 21600 will generate the same result.

This will process data valid on 2019-02-01 at 00Z, 06Z, 12Z, and 18Z as well as 2019-02-02 at 00Z. For each of these valid times, the METplus wrappers can also loop over a set of forecast leads that are all valid at the current run time. See *Looping over Forecast Leads* (page 50) for more information.

5.5.1.2.5 VALID_LIST

If the intervals between run times are irregular, then an explicit list of times can be defined. The following example will process the same times as the previous example:

```
[config]
LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_LIST = 2019020100, 2019020106, 2019020112, 2019020118, 2019020200
```

See the glossary entry for *VALID_LIST* for more information.

5.5.1.3 Looping by Initialization Time

When looping over initialization time (*LOOP_BY* = INIT or LOOP_BY = RETRO), the following variables must be set:

5.5.1.3.1 INIT_TIME_FMT

This is the format of the initialization times the user can configure in METplus Wrappers. The value of *INIT_BEG* and *INIT_END* must correspond to this format. Example: INIT_TIME_FMT = %Y%m%d%H. Using this format, the initialization time range values specified must be defined as YYYYMMDDHH, i.e. 2019020112.

5.5.1.3.2 INIT_BEG

This is the first initialization time that will be processed. The format of this variable is controlled by *INIT_TIME_FMT*. For example, if INIT_TIME_FMT = %Y%m%d, then INIT_BEG must be set to an initialization time matching YYYYMMDD, such as 20190201.

5.5.1.3.3 INIT_END

This is the last initialization time that can be processed. The format of this variable is controlled by INIT_TIME_FMT. For example, if INIT_TIME_FMT = %Y%m%d, then INIT_END must be set to an initialization time matching YYYYMMDD, such as 20190202.

Note: The time specified for this variable will not necessarily be processed. It is used to determine the cutoff of run times that can be processed. For example, if METplus Wrappers is configured to start at 2019-02-01 and end at 2019-02-02 processing data in 48 hour increments, it will process 2019-02-01 then increment the run time to 2019-02-03. This is later than the INIT_END valid, so execution will stop. However, if the increment is set to 24 hours (see INIT_INCREMENT), then METplus Wrappers will process initialization times 2019-02-01 and 2019-02-02 before ending executaion.

5.5.1.3.4 INIT_INCREMENT

This is the time interval to add to each run time to determine the next run time to process. See *Time Interval Units* (page 52) for information on time interval formatting. Units of hours are assumed if no units are specified. This value must be greater than or equal to 60 seconds because the METplus wrappers currently do not support processing intervals of less than one minute.

The following is a configuration that will process initialization time 2019-02-01 at 00Z until 2019-02-02 at 00Z in 6 hour (21600 second) increments:

```
[config]
LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2019020100
INIT_END = 2019020200
INIT_INCREMENT = 6H
```

Note: Substituting VALID_INCREMENT = 21600 will generate the same result.

This will process data initialized on 2019-02-01 at 00Z, 06Z, 12Z, and 18Z as well as 2019-02-02 at 00Z. For each of these initialization times, METplus Wrappers can also loop over a set of forecast leads that are all initialized at the current run time. See *Looping over Forecast Leads* (page 50) for more information.

5.5.1.3.5 INIT_LIST

If the intervals between run times are irregular, then an explicit list of times can be defined. The following example will process the same times as the previous example:

```
[config]
LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_LIST = 2019020100, 2019020106, 2019020112, 2019020118, 2019020200
```

See the glossary entry for *INIT_LIST* for more information.

5.5.1.4 Looping over Forecast Leads

Many of the wrappers will also loop over a list of forecast leads relative to the current valid/initialization time that is being processed.

5.5.1.4.1 LEAD_SEQ

This variable can be set to a comma-separated list of integer values (with optional units) to define the forecast leads that will be processed relative to the initialization/valid time. See *Time Interval Units* (page 52) for information on time interval formatting. Units of hours are assumed if no units are specified. For example:

```
[config]
LEAD_SEQ = 3, 6, 9
```

If *LOOP_BY* = VALID and the current run time is 2019-02-01 at 00Z, then three times will be processed:

1. Initialized on 2019-01-31 at 21Z / valid on 2019-02-01 at 00Z
2. Initialized on 2019-01-31 at 18Z / valid on 2019-02-01 at 00Z
3. Initialized on 2019-01-31 at 15Z / valid on 2019-02-01 at 00Z

If *LOOP_BY* = INIT and the current run time is 2019-02-01 at 00Z, then three times will be processed:

1. Initialized on 2019-02-01 at 00Z / valid on 2019-02-01 at 03Z
2. Initialized on 2019-02-01 at 00Z / valid on 2019-02-01 at 06Z
3. Initialized on 2019-02-01 at 00Z / valid on 2019-02-01 at 09Z

The user can also define *LEAD_SEQ* using *Begin End Increment (begin_end_incr)* (page 73) for many forecast leads. For example:

```
[config]
LEAD_SEQ = begin_end_incr(0,12,3)
```

is equivalent to setting:

```
[config]
LEAD_SEQ = 0, 3, 6, 9, 12
```

Grouping forecast leads is possible as well using a special version of the *LEAD_SEQ* variable for the **SeriesByLead Wrapper Only**. If *SERIES_BY_LEAD_GROUP_FCSTS* = True, then groups of forecast leads can be defined to be evaluated together. Any number of these groups can be defined by setting configuration variables LEAD_SEQ_1, LEAD_SEQ_2, . . . , *LEAD_SEQ_<n>*. The value can be defined with a comma-separated list of integers (currently only hours are supported here) or using *Begin End Increment (begin_end_incr)* (page 73). Each *LEAD_SEQ_<n>* must have a corresponding variable *LEAD_SEQ_<n>_LABEL*. For example:

```
[config]
LEAD_SEQ_1 = 0, 6, 12, 18
LEAD_SEQ_1_LABEL = Day1
LEAD_SEQ_2 = begin_end_incr(24,42,6)
LEAD_SEQ_2_LABEL = Day2
```

5.5.1.4.2 INIT_SEQ

If METplus Wrappers is configured to loop by valid time (*LOOP_BY* = VALID), INIT_SEQ can be used instead of *LEAD_SEQ*. This is a list of initialization hours that are available in the data. This is useful if the data initialization times are known and a different list of forecast leads should be used depending on the valid time being evaluated. For example:

```
[config]
LOOP_BY = VALID
INIT_SEQ = 0, 6, 12, 18
```

At valid time 2019-02-01 00Z, this initialization sequence will build a forecast lead list of 0, 6, 12, 18, 24, 30, etc. and at valid time 2019-02-01 01Z, this initialization sequence will build a forecast lead list of 1, 7, 13, 19, 25, 31, etc.

If using *INIT_SEQ*, restrict the forecast leads that will be used by setting *LEAD_SEQ_MIN* and *LEAD_SEQ_MAX*. For example, to only process forecast leads between 12 and 24 set:

```
[config]
LEAD_SEQ_MIN = 12
LEAD_SEQ_MAX = 24
```

At valid time 2019-02-01 00Z, this initialization sequence will build a forecast lead list of 12, 18, 24 and at valid time 2019-02-01 01Z, this initialization sequence will build a forecast lead list of 13, 19.

Setting minimum and maximum values will also affect the list of forecast leads if *LEAD_SEQ* is used. *LEAD_SEQ* takes precedence over *INIT_SEQ*, so if both variables are set in the configuration, *INIT_SEQ* will be ignored in favor of *LEAD_SEQ*.

5.5.1.5 Time Interval Units

Time intervals defined in configuration variables each have default values: *LEAD_SEQ* and *INIT_SEQ* default to hours, *VALID_INCREMENT* and *INIT_INCREMENT* default to seconds. Units of years, months, days, hours, minutes, or seconds can also be specified by adding a letter (Y, m, d, H, M, or S respectively) to the end of the number. If no units are specified, seconds are assumed.

Examples:

```
3600 : 3600 seconds
3600S : 3600 seconds
60M : 60 minutes or 3600 seconds
1H : 1 hour or 3600 seconds
1m : 1 month (relative)
1d : 1 day or 24 hours or 86400 seconds
1Y : 1 year (relative)
```

Units of months (m) and years (Y) do not have set intervals because the length of a month or year is relative to the relative date/time. Therefore these intervals are calculated based on the current run time and cannot be expressed in seconds unless the run time value is available.

5.5.1.6 Skipping Times

Version 3.1 added the ability to skip certain valid times. The configuration variable *SKIP_TIMES* can be used to provide a list of time formats each with a list of times to not process. The format and time list are separated by a colon. Any numeric python strftime formatting directive can be used as the time format (see https://strftime.org). Each item in the list must be surrounded by quotation marks. Here are a few examples.

Example 1:

```
[config]
SKIP_TIMES = "%m:3"
```

This will skip the 3rd month, March.

Example 2:

```
[config]
SKIP_TIMES = "%d:30,31"
```

This will skip every 30th and 31st day.

Example 3:

```
[config]
SKIP_TIMES = "%d:30,31", "%m:3"
```

This will skip every 30th and 31st day **and** every 3rd month.

Begin End Increment (begin_end_incr) (page 73) syntax can be used to define a range of times to skip.

b = begin value, e = end value,

i = increment between each value

Example 4:

```
[config]
SKIP_TIMES = "%H:begin_end_incr(0,22,2)"
```

This will skip every even hour (starting from 0, ending on 22, by 2). This is equivalent to:

```
[config]
SKIP_TIMES = "%H:0,2,4,6,8,10,12,14,16,18,20,22"
```

Multiple strftime directives can be specified in a single time format.

Example 5:

```
[config]
SKIP_TIMES = "%Y%m%d:19991231, 20141031"
```

This will skip the dates Dec. 31, 1999 and Oct. 31, 2014.

To only skip certain times for a single wrapper, use a wrapper-specific variable. Using a wrapper-specific variable will ignore the generic SKIP_TIMES values.

Example 6:

```
[config]
GRID_STAT_SKIP_TIMES = "%m:3,4,5,6,7,8,9,10,11"
SKIP_TIMES = "%d:31"
```

This will skip the months March through November for GridStat wrapper only. All other wrappers in the *PROCESS_LIST* will skip the 31st day of each month. Note that the SKIP_TIMES values are not applied to GridStat in this case.

5.5.1.7 Realtime Looping

5.5.1.7.1 Now and Today

To make running in realtime easier, the METplus Wrappers support defining the begin and end times relative to the current clock time. For example, if the current time is 2019-04-26 08:17 and the METplus Wrappers is run with:

```
[config]
VALID_END = {now?fmt=%Y%m%d%H}
```

then the value of *VALID_END* will be set to 2019042608. {today} can also be used to substitute the current YYYYMMDD, i.e. 20190426. The formatting for the 'today' keyword cannot be changed..

5.5.1.7.2 Shift Keyword

The 'shift' keyword can be used to shift the current time by any number of seconds. For example, if the METplus Wrappers are run at the same clock time with:

```
[config]
VALID_BEG = {now?fmt=%Y%m%d%H?shift=-86400}
```

then the value of *VALID_BEG* will be set to the current clock time shifted by -86400 seconds (24 hours backwards), or 2019-04-25 08Z.

The value defined for 'shift' also supports *Time Interval Units* (page 52).

If *VALID_INCREMENT* is set to 21600 seconds (6 hours), then the METplus Wrappers will process the following valid times:

2019-04-25 08Z
2019-04-25 14Z
2019-04-25 20Z
2019-04-26 02Z
2019-04-26 08Z

5.5.1.7.3 Truncate Keyword

The user may want to configure the METplus Wrappers to process at 00Z, 06Z, 12Z, and 18Z of a given day instead of 02Z, 08Z, 14Z, and 20Z. Having to adjust the shift amount differently if running at 08Z or 09Z to get the times to line up would be tedious. Instead, use the 'truncate' keyword. The value set here is the number of seconds that is used to determine the interval of time to round down. To process every 6 hours, set 'truncate' to 21600 seconds:

```
[config]
VALID_BEG = {now?fmt=%Y%m%d%H?shift=-86400?truncate=21600}
```

This will round down the value to the nearest 6 hour interval of time. Starting METplus Wrappers on or after 06Z but before 12Z on 20190426 will result in VALID_BEG = 2019042506 (clock time shifted backwards by 24 hours then truncated to the nearest 6 hour time).

Starting METplus Wrappers on 20190426 at 08:16 with the following configuration:

```
[config]
LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = {now?fmt=%Y%m%d%H?shift=-86400?truncate=21600}
VALID_END = {now?fmt=%Y%m%d%H}
VALID_INCREMENT = 21600
```

will process valid times starting on 20190425 at 06Z every 6 hours until the current run time is later than 20190426 at 08Z, which will result in processing the following valid times:

20190425_06
20190425_12
20190425_18
20190426_00
20190426_06

Note: When using the 'now' keyword, the value of VALID_TIME_FMT must be identical to the 'fmt' value corresponding to the 'now' item in VALID_BEG and VALID_END. In the above example, this would be the %Y%m%d%H portion within values of the VALID_TIME_FMT, VALID_BEG, and VALID_END variables.

5.5.2 Process List

The PROCESS_LIST variable defines the list of wrappers to run. This can be a single value or a comma separated list of values. Each value must match an existing wrapper name without the 'Wrapper' suffix.

Example 1 Configuration:

```
[config]
PROCESS_LIST = GridStat
```

This example will run GridStatWrapper only.

Example 2 Configuration:

```
[config]
PROCESS_LIST = PCPCombine, GridStat
```

This example will run PCPCombineWrapper then GridStatWrapper.

5.5.2.1 Instance Names in Process List

Added in version 4.0.0 is the ability to specify an instance name for each process in the PROCESS_LIST. This allows multiple instances of the same wrapper to be specified in the PROCESS_LIST. Users can create a new section header in their configuration files with the same name as the instance. If defined, values in this section will override the values in the configuration for that instance. The instance name of the process is defined by adding text after the process name inside parenthesis. There should be no space between the process name and the parenthesis.

Example 3 Configuration:

```
[config]
PROCESS_LIST = GridStat, GridStat(my_instance_name)

GRID_STAT_OUTPUT_DIR = /grid/stat/output/dir

[my_instance_name]
GRID_STAT_OUTPUT_DIR = /my/instance/name/output/dir
```

In this example, the first occurrence of GridStat in the PROCESS_LIST does not have an instance name associated with it, so it will use the value /grid/stat/output/dir as the output directory. The second occurrence has an instance name 'my_instance_name' and there is a section header with the same name, so this instance will use /my/instance/name/output/dir as the output directory.

5.5.3 Loop Order

The METplus wrappers will run all times for the first process defined in the *PROCESS_LIST*, then run all times for the second process, and so on. The *LOOP_ORDER* variable has been deprecated in v5.0.0. This is the behavior that was previously executed when LOOP_ORDER = processes.

Example Configuration:

```
[config]

PROCESS_LIST = PCPCombine, GridStat

VALID_BEG = 20190201
VALID_END = 20190203
VALID_INCREMENT = 1d
```

will run in the following order:

```
* PCPCombine at 2019-02-01
* PCPCombine at 2019-02-02
* PCPCombine at 2019-02-03
* GridStat   at 2019-02-01
* GridStat   at 2019-02-02
* GridStat   at 2019-02-03
```

5.5.4 Custom Looping

A list of text strings can be defined in the METplus wrappers configuration files to allow each wrapper to process data multiple times for each run time. The strings can be referenced in various places in the METplus configuration files to change input/output file paths, configuration file paths, and more. The value of each list item can be referenced in the METplus configuration variables by using {custom?fmt=%s}. The variable CUSTOM_LOOP_LIST will apply the values to each wrapper in the PROCESS_LIST unless the wrapper does not support this functionality. CyclonePlotter, StatAnalysis, TCStat, and TCMPRPlotter wrappers are not supported. If the variable is not set or set to an empty string, the wrapper will execute as normal without additional runs. The name of the wrapper-specific variables contain the name of the wrapper, i.e. SERIES_ANALYSIS_CUSTOM_LOOP_LIST, PCP_COMBINE_CUSTOM_LOOP_LIST, GRID_STAT_CUSTOM_LOOP_LIST, etc. Setting these variables will override the value set for CUSTOM_LOOP_LIST for that wrapper only.

Example 1 Configuration (Reading different input files):

```
[config]
PROCESS_LIST = PCPCombine

VALID_BEG = 20190201
VALID_END = 20190203
VALID_INCREMENT = 1d

PCP_COMBINE_CUSTOM_LOOP_LIST = mem_001, mem_002

FCST_PCP_COMBINE_INPUT_DIR = /d1/ensemble

FCST_PCP_COMBINE_INPUT_TEMPLATE = {custom?fmt=%s}/{valid?fmt=%Y%m%d}.nc
```

This configuration will run the following:

- PCPCombine at 2019-02-01 reading from /d1/ensemble/mem_001/20190201.nc
- PCPCombine at 2019-02-01 reading from /d1/ensemble/mem_002/20190201.nc
- PCPCombine at 2019-02-02 reading from /d1/ensemble/mem_001/20190202.nc
- PCPCombine at 2019-02-02 reading from /d1/ensemble/mem_002/20190202.nc
- PCPCombine at 2019-02-03 reading from /d1/ensemble/mem_001/20190203.nc
- PCPCombine at 2019-02-03 reading from /d1/ensemble/mem_002/20190203.nc

Example 2 Configuration (Using different MET config files):

```
[config]
PROCESS_LIST = SeriesAnalysis

VALID_BEG = 20190201
VALID_END = 20190203
VALID_INCREMENT = 1d

SERIES_ANALYSIS_CUSTOM_LOOP_LIST = one, two

SERIES_ANALYSIS_CONFIG_FILE = {CONFIG_DIR}/SAConfig_{custom?fmt=%s}

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/SA/{custom?fmt=%s}
```

This configuration will run SeriesAnalysis:

- At 2019-02-01 using SAConfig_one config file and writing output to {OUTPUT_BASE}/SA/one
- At 2019-02-01 using SAConfig_two config file and writing output to {OUTPUT_BASE}/SA/two
- At 2019-02-02 using SAConfig_one config file and writing output to {OUTPUT_BASE}/SA/one
- At 2019-02-02 using SAConfig_two config file and writing output to {OUTPUT_BASE}/SA/two
- At 2019-02-03 using SAConfig_one config file and writing output to {OUTPUT_BASE}/SA/one
- At 2019-02-03 using SAConfig_two config file and writing output to {OUTPUT_BASE}/SA/two

5.5.5 Field Info

This section describes how METplus Wrappers configuration variables can be used to define field information that is sent to the MET applications to read forecast and observation fields.

5.5.5.1 FCST_VAR<n>_NAME

Set this to the name of a forecast variable to evaluate. <n> is any integer greater than or equal to 1, i.e.:

```
[config]
FCST_VAR1_NAME = TMP
FCST_VAR2_NAME = RH
```

If this value is set for a given <n> value, then the corresponding OBS_VAR<n>_NAME must be set. If the value for forecast and observation data are the same, BOTH_VAR<n>_NAME can be used instead.

5.5.5.2 FCST_VAR<n>_LEVELS

Set this to a comma-separated list of levels or a single value. FCST_VAR1_LEVELS corresponds to FCST_VAR1_NAME, FCST_VAR2_LEVELS corresponds to FCST_VAR2_NAME, etc. For example:

```
[config]
FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = P500, P750
```

will process TMP at P500 and TMP at P750. If FCST_VAR<n>_LEVELS and FCST_VAR<n>_NAME are set, then the corresponding OBS_VAR<n>_LEVELS and OBS_VAR<n>_NAME must be set. If the value for forecast and observation data are the same, BOTH_VAR<n>_NAME and BOTH_VAR<n>_LEVELS can be used instead.

5.5.5.3 OBS_VAR<n>_NAME

Set this to the corresponding observation variable to evaluate with FCST_VAR<n>_NAME. If this value is set for a given <n> value, then the corresponding FCST_VAR<n>_NAME must be set. If the value for forecast and observation data are the same, BOTH_VAR<n>_NAME can be used instead.

5.5.5.4 OBS_VAR<n>_LEVELS

Set this to a comma-separated list of levels or a single value. If OBS_VAR<n>_LEVELS and OBS_VAR<n>_NAME are set, then the corresponding FCST_VAR<n>_LEVELS and FCST_VAR<n>_NAME must be set. If the value for forecast and observation data are the same, BOTH_VAR<n>_NAME and BOTH_VAR<n>_LEVELS can be used instead. For example, setting:

```
[config]
BOTH_VAR1_NAME = TMP
BOTH_VAR1_LEVELS = P500
BOTH_VAR2_NAME = RH
BOTH_VAR2_LEVELS = P750, P250
```

is the equivalent of setting:

```
[config]
FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = P500
FCST_VAR2_NAME = RH
FCST_VAR2_LEVELS = P750, P250
OBS_VAR1_NAME = TMP
OBS_VAR1_LEVELS = P500
OBS_VAR2_NAME = RH
OBS_VAR2_LEVELS = P750, P250
```

This will compare:

TMP/P500 in the forecast data to TMP/P500 in the observation data
RH/P750 in the forecast data to RH/P750 in the observation data
RH/P250 in the forecast data to RH/P250 in the observation data

If setting:

```
[config]
FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = P500, P750
OBS_VAR1_NAME = TEMP
OBS_VAR1_LEVELS = "(0,*,*)","(1,*,*)"
```

METplus Wrappers will compare:

TMP/P500 in the forecast data to TEMP at (0,*,*) in the observation data
TMP/P750 in the forecast data to TEMP at (1,*,*) in the observation data

---

Note: NetCDF level values that contain (*,*) notation must be surrounded by quotation marks so it will not be misinterpreted as a list of items.

The number of level items must be equal in each list for a given comparison. If separate names for a forecast and observation are defined, separate levels must be defined for each even if they are equivalent. For example, setting FCST_VAR1_NAME, FCST_VAR1_LEVELS, and OBS_VAR1_NAME, but not setting OBS_VAR1_LEVELS will result in an error.

The field information specified using the *_NAME/*_LEVELS variables will be formatted to match the field info dictionary in the MET config files and passed to the appropriate config file to evaluate the data. The previous configuration comparing TMP (P500 and P750) in the forecast data and TEMP ((0,*,*)) in the observation data will generate the following in the MET config file:

```
fcst = {field = [ {name="TMP"; level="P500";} ];}
obs = {field = [{name="TEMP"; level="(0,*,*)";} ];}
```

and then comparing TMP (P500 and P750) in the forecast data and TEMP ((1,*,*)) in the observation data will generate the following in the MET config file:

```
fcst = {field = [ {name="TMP"; level="P750";} ];}
obs = {field = [{name="TEMP"; level="(1,*,*)";} ];}
```

Note that some MET applications allow multiple fields to be specified for a single run. If the MET tool allows it and METplus Wrappers is configured accordingly, these two comparisons would be configured in a single run.

5.5.5.5 Read explicit time dimension from a NetCDF level

If the input NetCDF data contains a time dimension, the time can be specified in the level value. The MET tool will find the data for the time requested:

```
[config]
OBS_VAR1_NAME = TEMP
OBS_VAR1_LEVELS = "(20190201_120000,*,*)"
```

This example will extract the data that corresponds to Feb. 1, 2019 at 12Z if it is available (see the MET Documentation for more information on this functionality). The time can be specified based on the current run time, i.e.:

```
[config]
OBS_VAR1_NAME = TEMP
OBS_VAR1_LEVELS = "({valid?fmt=%Y%m%d_%H%M%S},*,*)"
```

In this example, {valid?fmt=%Y%m%d_%H%M%S} will be substituted with the valid time of the current run.

5.5.5.6 Substituting Current Level

When using Python Embedding to pass in data for a field, one may want to call the same Python script for each vertical level specifying the level string for each call. In this case, a list of levels can be specified using *FCST_VAR<n>_LEVELS* and the value can be substituted into the corresponding *FCST_VAR<n>_NAME* using {fcst_level}:

```
[config]
FCST_VAR1_NAME = {INPUT_BASE}/myscripts/read_nc2xr.py {INPUT_BASE}/mydata/forecast_file.nc4␣
↪TMP {valid?fmt=%Y%m%d_%H%M} {fcst_level}
FCST_VAR1_LEVELS = P1000,P850,P700,P500,P250,P100
```

This will call the Python script 6 times:

- {INPUT_BASE}/myscripts/read_nc2xr.py {INPUT_BASE}/mydata/forecast_file.nc4 TMP {valid?fmt=%Y%m%d_%H%M} P1000
- {INPUT_BASE}/myscripts/read_nc2xr.py {INPUT_BASE}/mydata/forecast_file.nc4 TMP {valid?fmt=%Y%m%d_%H%M} P850
- {INPUT_BASE}/myscripts/read_nc2xr.py {INPUT_BASE}/mydata/forecast_file.nc4 TMP {valid?fmt=%Y%m%d_%H%M} P700
- {INPUT_BASE}/myscripts/read_nc2xr.py {INPUT_BASE}/mydata/forecast_file.nc4 TMP {valid?fmt=%Y%m%d_%H%M} P500
- {INPUT_BASE}/myscripts/read_nc2xr.py {INPUT_BASE}/mydata/forecast_file.nc4 TMP {valid?fmt=%Y%m%d_%H%M} P250
- {INPUT_BASE}/myscripts/read_nc2xr.py {INPUT_BASE}/mydata/forecast_file.nc4 TMP {valid?fmt=%Y%m%d_%H%M} P100

This only applies if the wrapper runs once per field name/level combination such as MODE or if the wrapper is configured to do so, for example GridStat using *GRID_STAT_ONCE_PER_FIELD*.

The same logic applies for observation data using *OBS_VAR<n>_NAME*, *OBS_VAR<n>_LEVELS*, and {obs_level}.

To reference the current field name and/or level in another configuration variable such as *MODE_OUTPUT_PREFIX*, use {CURRENT_FCST_NAME}, {CURRENT_FCST_LEVEL}, {CURRENT_OBS_NAME}, and/or {CURRENT_OBS_LEVEL}.

5.5.5.7 FCST_VAR<n>_THRESH / OBS_VAR<n>_THRESH

Set this to a comma-separated list of threshold values to use in the comparison. Each of these values must begin with a comparison operator (>, >=, =, ==, !=, <, <=, gt, ge, eq, ne, lt, or le). For example, setting:

```
[config]
FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = P500
FCST_VAR1_THRESH = le0.5, gt0.4, gt0.5, gt0.8
```

will add the following information to the MET config file:

```
fcst = {field = [ {name="TMP"; level="P500"; cat_thresh=[ le0.5, gt0.4, gt0.5, gt0.8];} ];}
```

If FCST_VAR<n>_THRESH is set, then OBS_VAR<n>_THRESH must be set. If the threshold list is the same for both forecast and observation data, BOTH_VAR<n>_THRESH can be used instead.

5.5.5.8 FCST_VAR<n>_OPTIONS / OBS_VAR<n>_OPTIONS

Set this to add additional information to the field dictionary in the MET config file. The item must end with a semi-colon. For example:

```
[config]
FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = P500
FCST_VAR1_OPTIONS = GRIB_lvl_typ = 105; ens_phist_bin_size = 0.05;
```

will add the following to the MET config file:

```
fcst = {field = [ {name="TMP"; level="P500"; GRIB_lvl_typ = 105; ens_phist_bin_size = 0.05;}␣
↪];}
```

If FCST_VAR<n>_OPTIONS is set, OBS_VAR<n>_OPTIONS does not need to be set, and vice-versa. If the extra options are the same for both forecast and observation data, BOTH_VAR<n>_OPTIONS can be used instead.

ENS_VAR<n>_NAME / *ENS_VAR<n>_LEVELS*/ *ENS_VAR<n>_THRESH* / *ENS_VAR<n>_OPTIONS*: **Used with EnsembleStat Wrapper only.** Users may want to define the ens dictionary item in the MET EnsembleStat config file differently than the fcst dictionary item. If this is the case, then use these variables. If it is

not set, the values in the corresponding FCST_VAR<n>_[NAME/LEVELS/THRESH/OPTIONS] will be used in the ens dictionary.

5.5.5.9 Probabilistic Forecast Fields

If processing probabilistic forecast fields, there are additional configuration variables that are used to properly format the field info that is passed into the wrapped MET configuration files. *FCST_IS_PROB* is used to process probabilistic data:

```
[config]
FCST_IS_PROB = True
FCST_VAR1_NAME = APCP_24_A24_ENS_FREQ_gt0.0
FCST_VAR1_LEVELS = "(*,*)"
```

will add the following to the MET config file:

```
fcst = {field = [{ name="APCP_24_A24_ENS_FREQ_gt0.0"; level="(*,*)"; prob=TRUE; cat_thresh=[␣
↪==0.1 ]; }];}
```

The cat_thresh value defaults to ==0.1 and defines the size of the Nx2 probabilistic contingency table. It is set by *FCST_GRID_STAT_PROB_THRESH* (for GridStat):

```
[config]
FCST_IS_PROB = True
FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = "(*,*)"
FCST_GRID_STAT_PROB_THRESH = ==0.2
```

will add the following to the MET config file:

```
fcst = {field = [{ name="APCP"; level="(*,*)"; prob=TRUE; cat_thresh=[ ==0.2 ]; }];}
```

Some GRIB files contain probabilistic field information in the Product Definition Section (PDS). The format of the fcst.field info to read these data expect the name to be set to "PROB" and the field name/level values are set inside a prob dictionary. If this is the case, then *FCST_PROB_IN_GRIB_PDS* should be set to True. At least 1 threshold must be set with *FCST_VAR<n>_THRESH* in this case. The threshold value will be formatted in the prob dictionary using thresh_lo and/or thresh_hi values:

```
[config]
FCST_IS_PROB = True
FCST_PROB_IN_GRIB_PDS = True
FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A03
FCST_VAR1_THRESH = gt12.7
```

will add the following to the MET config file:

```
fcst = {field = [{ name="PROB"; level="A03"; prob={ name="APCP"; thresh_lo=12.7; } cat_
↪thresh=[ ==0.1 ]; }];}
```

5.5.5.10 Wrapper Specific Field Info

New to METplus 3.0 is the ability to specify VAR<n> items differently across comparison wrappers. In previous versions, it was assumed that the list of forecast and observation files that were processed would be applied to any MET Stat tool used, such as GridStat, PointStat, EnsembleStat, MODE, or MTD. This prevented the ability to run, for example, EnsembleStat, then pass the output into GridStat.

Example 1:

```
[config]
PROCESS_LIST = EnsembleStat, GridStat

FCST_ENSEMBLE_STAT_VAR1_NAME = HGT
FCST_ENSEMBLE_STAT_VAR1_LEVELS = P500

FCST_GRID_STAT_VAR1_NAME = HGT_P500_ENS_MEAN
FCST_GRID_STAT_VAR1_LEVELS = "(*,*)"
```

If the generic *FCST_VAR<n>_NAME* variables are used, the same values will be applied to all tools that don't have wrapper specific fields defined. If wrapper specific fields are defined, any generic fields will be ignored.

Example 2:

```
[config]
PROCESS_LIST = GridStat, EnsembleStat

FCST_VAR1_NAME = HGT
FCST_VAR1_LEVELS = P500, P750
FCST_VAR2_NAME = TMP
FCST_VAR2_LEVELS = P500, P750

FCST_ENSEMBLE_STAT_VAR1_NAME = HGT
FCST_ENSEMBLE_STAT_VAR1_LEVELS = P500
```

In this example, GridStat will process HGT at pressure levels 500 and 750 and TMP at pressure levels 500 and 750, while EnsembleStat will only process HGT at pressure level 500. To configure EnsembleStat to also process TMP, the user will have to define it explicitly with FCST_ENSEMBLE_STAT_VAR2_NAME.

This functionality applies to GridStat, EnsembleStat, PointStat, MODE, and MTD wrappers only.

For more information on GRIB_lvl_typ and other file-specific commands, review the MET User's Guide, Chapter 3.

5.5.6 Directory and Filename Template Info

The METplus Wrappers use directory and filename template configuration variables to find the desired files for a given run.

5.5.6.1 Using Templates to find Observation Data

The following configuration variables describe input observation data:

```
[config]
OBS_GRID_STAT_INPUT_DIR = /my/path/to/grid_stat/input/obs

OBS_GRID_STAT_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/prefix.{valid?fmt=%Y%m%d%H}.ext
```

The input directory is the top level directory containing all of the observation data. The template contains items with keywords that will be substituted with time values for each run. After the values are substituted, METplus Wrappers will check to see if the desired file exists relative to the input directory. At valid time 20190201_12Z, the full desired path of the observation input data to grid_stat will be:

/my/path/to/grid_stat/input/obs/20190201/prefix.2019020112.ext

Note that the template contains a dated subdirectory. This cannot go in the OBS_GRID_STAT_INPUT_DIR variable because the dated subdirectory changes based on the run time.

METplus Wrappers does not need to be configured to loop by valid time to find files using a template containing valid time information. For example, at initialization time 20190201_12Z and forecast lead 3, the valid time is calculated to be 20190201_15Z and the full desired path of the observation input data to grid_stat will be:

/my/path/to/grid_stat/input/obs/20190201/prefix.2019020115.ext

The 'init' and 'valid' are keywords used to denote initialization and valid times respectively. Other keywords that are supported include 'lead', 'offset', 'da_init', and 'cycle' which can all be used to find forecast data and data assimilation data depending on the task.

5.5.6.2 Using Templates to find Forecast Data

Most forecast files contain the initialization time and the forecast lead in the filename. The keywords 'init' and 'lead' can be used to describe the template of these files:

```
[config]
FCST_GRID_STAT_INPUT_DIR = /my/path/to/grid_stat/input/fcst

FCST_GRID_STAT_INPUT_TEMPLATE = prefix.{init?fmt=%Y%m%d%H}_f{lead?fmt=%3H}.ext
```

For a valid time of 20190201_00Z and a forecast lead of 3, METplus Wrappers will look for the following forecast file:

/my/path/to/grid_stat/input/fcst/prefix.2019013121_f003.ext

Some forecast file names contain the forecast lead time in seconds, padded with zeros. In this case, the 'lead' keyword with the format (fmt) set to %8S will use the forecast lead seconds with 8 digits as shown below:

```
[config]
FCST_GRID_STAT_INPUT_DIR = /my/path/to/grid_stat/input/fcst

FCST_GRID_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/g_{init?fmt=%H%M%S}/f_{lead?fmt=%8S}.ext
```

For a valid time of 20190201_03Z and a forecast lead of 3 hours, METplus Wrappers will look for the following forecast file:

/my/path/to/grid_stat/input/fcst/20190201/g_000000/f_00010800.ext

5.5.6.3 Using Templates to find Data Assimilation Data

Some data assimilation files contain offset and da_init (data assimilation initialization) values in the filename. These values are used to determine the valid time of the data. Consider the following configuration:

```
[config]
PB2NC_OFFSETS = 6, 3

PB2NC_INPUT_DIR = /my/path/to/prepbufr

PB2NC_INPUT_TEMPLATE = prefix.{da_init?fmt=%Y%m%d}_{cycle?fmt=%H}_off{offset?fmt=%2H}.ext
```

The PB2NC_OFFSETS list tells METplus Wrappers the order in which to prioritize files with offsets in the name. At valid time 20190201_12Z, METplus Wrappers will check if the following file exists:

/my/path/to/prepbufr/prefix.20190201_18_off06.ext

The offset is added to the valid time to get the data assimilation initialization time. Note that 'cycle' can be used interchangeably with 'da_init'. It is generally used to specify the hour of the data that was generated. If that file doesn't exist, it will check if the following file exists:

/my/path/to/prepbufr/prefix.20190201_15_off03.ext

5.5.6.4 Shifting Times in Filename Templates

Users can use the 'shift' keyword to adjust the time referenced in the filename template relative to the run time. For example, if the input files used contained data from 01Z on the date specified in the filename to 01Z on the following day. In this example, for a run at 00Z use the file from the previous day and for the 01Z to 23Z runs, use the file that corresponds to the current day. Here is an example:

```
[config]
OBS_POINT_STAT_INPUT_TEMPLATE = {valid?fmt=%Y%m%d?shift=-3600}.ext
```

Running the above configuration at a valid time of 20190201_12Z will shift the valid time backwards by 1 hour (3600 seconds) resulting in 20190201_11Z and will substitute the current day into the template, giving a filename of 20190201.ext. Running at valid time 20190201_00Z, the shift will result in a file time of 20190131_23Z, so the filename will be 20190131.ext that is generated by the template.

5.5.6.5 Multiple Input Files

If a tool supports reading multiple files for a given input, then there are a variety of ways to configure the METplus wrappers to read them.

- *Wildcards* (page 67)
- *List of Templates* (page 68)
- *File Windows* (page 68)

5.5.6.5.1 Using Wildcards to find Multiple Files

Wildcard characters can be used in filename template variables. The * character is used to match 1 or more characters and the ? character is used to match a single character.

For example, if a directory /my/files contains the following files:

- filename_AAA.nc
- filename_ABA.nc

- filename_BBB.nc
- filename.nc

The following template will match filename_AAA.nc, filename_ABA.nc and filename_BBB.nc, but not filename.nc:

```
INPUT_TEMPLATE = /my/files/filename_*.nc
```

The following template will match filename_AAA.nc and filename_ABA.nc:

```
INPUT_TEMPLATE = /my/files/filename_A?A.nc
```

5.5.6.5.2 Using a List of Templates to find Multiple Files

A comma-separated list of templates can be specified in a _TEMPLATE variable. Each value in the list will be added to the corresponding _DIR variable.

For example, if a directory /my/files contains the following files:

- filename_AAA.nc
- filename_ABA.nc
- filename_BBB.nc
- filename.nc

The following configuration will look for files /my/files/filename_AAA.nc and /my/files/filename_BBB.nc:

```
INPUT_DIR = /my/files
INPUT_TEMPLATE = filename_AAA.nc, filename_BBB.nc
```

Lists of templates can be used with *wildcards* (page 67). The following configuration will find all 4 files in /my/files:

```
INPUT_DIR = /my/files
INPUT_TEMPLATE = filename.nc, filename_*.nc
```

The *Begin End Increment (begin_end_incr)* (page 73) syntax can be used to generate lists of file paths.

5.5.6.5.3 Using File Windows to find Multiple Files

The *FCST_FILE_WINDOW_BEGIN*, *FCST_FILE_WINDOW_END*, *OBS_FILE_WINDOW_BEGIN*, and *OBS_FILE_WINDOW_END* configuration variables can be used if the time information in the input data does not exactly line up with the run time but the user still wants to process the data. The default value of the file window begin and end variables are both 0 seconds. If both values are set to 0, METplus Wrappers will require that a file matching the template with the exact time requested exists. If either value is non-zero, METplus Wrappers will examine all of the files under the input directory that match the

template, pull out the time information from the files, and use the file with the time closest to the run_time. For example, consider the following configuration:

```
[config]
OBS_FILE_WINDOW_BEGIN = -7200
OBS_FILE_WINDOW_END = 7200

OBS_GRID_STAT_INPUT_DIR = /my/grid_stat/input/obs

OBS_GRID_STAT_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/pre.{valid?fmt=%Y%m%d}_{valid?fmt=%H}.ext
```

For a run time of 20190201_00Z, and a set of files in the input directory that looks like this:

- /my/grid_stat/input/obs/20190131/pre.20190131_22.ext
- /my/grid_stat/input/obs/20190131/pre.20190131_23.ext
- /my/grid_stat/input/obs/20190201/othertype.20190201_00.ext
- /my/grid_stat/input/obs/20190201/pre.20190201_01.ext
- /my/grid_stat/input/obs/20190201/pre.20190201_02.ext

The following behavior can be expected for each file:

1. The first file matches the template and the file time is within the window, so the filename and time difference relative to the valid time (7200 seconds, or 2 hours) is saved.
2. The second file matches the template, the file time is within the window, and the time difference is less than the closest file so the filename and time difference relative to the valid time (3600 seconds, or 1 hour) is saved.
3. The third file does not match the template and is ignored.
4. The fourth file matches the template and is within the time range, but it is the same distance away from the valid time as the closest file. GridStat only allows one file to be processed so it is ignored (PB2NC is currently the only METplus Wrapper that allows multiple files to be processed).
5. The fifth file matches the template but it is a further distance away from the closest file (7200 seconds versus 3600 seconds) so it is ignored.

Therefore, /my/grid_stat/input/obs/20190131/pre.20190131_23.ext will be used as the input to grid_stat in this example.

Wrapper Specific Windows

A user may need to specify a different window on a wrapper-by-wrapper basis. If this is the case, the user can override the file window values for each wrapper. Consider the following configuration:

```
[config]
PROCESS_LIST = PCPCombine, GridStat, EnsembleStat
OBS_FILE_WINDOW_BEGIN = 0
OBS_FILE_WINDOW_END = 0
OBS_GRID_STAT_FILE_WINDOW_BEGIN = -1800
```

(continues on next page)

(continued from previous page)

```
OBS_GRID_STAT_FILE_WINDOW_END = 1800
OBS_ENSEMBLE_STAT_FILE_WINDOW_END = 3600
```

Using the above configuration, PCPCombine will use +/- 0 hours and require exact file times. GridStat will use -1800/+1800 for observation data and EnsembleStat will use -0/+3600 for observation data. *OBS_ENSEMBLE_STAT_FILE_WINDOW_BEGIN* was not set, so the EnsembleStat wrapper will use *OBS_FILE_WINDOW_BEGIN*.

5.5.7 Runtime Frequency

Some wrappers have an option to specify how frequently to process data. It can be run once to process all of the available files in the desired time range, or it can be configured to run over different intervals. This allows the user to aggregate the output in a variety of ways. The wrappers that support this functionality (along with the configuration variable that controls the setting) include:

- *SeriesAnalysis* (page 229) : *SERIES_ANALYSIS_RUNTIME_FREQ*
- *GridDiag* (page 144) : *GRID_DIAG_RUNTIME_FREQ*
- *UserScript* (page 307) : *USER_SCRIPT_RUNTIME_FREQ*

At the start of execution of the wrapper (SeriesAnalysis and GridDiag), a full list of all available files will be obtained. Then the wrapper will subset the data and call the MET tool based on the runtime frequency setting. UserScript wrapper will simply run at the interval specified without obtaining a list of files.

Depending on which option is selected, some filename template tags will translate to * when performing string substitution. The possible values for the *_RUNTIME_FREQ variables are:

- RUN_ONCE : Runs once processing all files. * is substituted for init/valid/lead
- RUN_ONCE_PER_INIT_OR_VALID : Run the command once for each initialization or valid time depending on the value of LOOP_BY. If LOOP_BY = INIT, * is substituted for valid and lead. If LOOP_BY = VALID, * is substituted for init and lead.
- RUN_ONCE_PER_LEAD : Run the command once for each forecast lead time. * is substituted for valid and init
- RUN_ONCE_FOR_EACH : Run the command once for every runtime (init or valid and forecast lead combination). All filename templates are substituted with values.

Note that the following example may not contain all of the configuration variables that are required for a successful run. The are intended to show how these variables affect how the data is processed.

SeriesAnalysis Examples:

```
[config]

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2020101712
INIT_END = 2020101912
```

(continues on next page)

(continued from previous page)

```
INIT_INCREMENT = 1d

LEAD_SEQ = 3H, 6H

PROCESS_LIST = SeriesAnalysis

FCST_SERIES_ANALYSIS_INPUT_DIR = /my/fcst/dir

FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = I{init?fmt=%Y%m%d%H}_F{lead?fmt=%3H}_V{valid?fmt=%H}
```

In this example, the wrapper will go through all initialization and forecast lead times and find any files that match the template under /my/fcst/dir:

Init: 2020-10-17 12Z, Forecast: 3 hour, File: I2020101712_F003_V15
Init: 2020-10-17 12Z, Forecast: 6 hour, File: I2020101712_F006_V18
Init: 2020-10-18 12Z, Forecast: 3 hour, File: I2020101812_F003_V15
Init: 2020-10-18 12Z, Forecast: 6 hour, File: I2020101812_F006_V18
Init: 2020-10-19 12Z, Forecast: 3 hour, File: I2020101912_F003_V15
Init: 2020-10-19 12Z, Forecast: 6 hour, File: I2020101912_F006_V18

Example 1: Run Once:

```
[config]
SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE
```

For this configuration, a single command will be built to call SeriesAnalysis. The wildcard character '*' will replace init, valid, and lead in the template when attempting to find data to process.

Template Used: I*_F*_V* Files Processed:

```
I2020101712_F003_V15
I2020101712_F006_V18
I2020101812_F003_V15
I2020101812_F006_V18
I2020101912_F003_V15
I2020101912_F006_V18
```

Example 2 Run Once Per Initialization Time:

```
[config]
SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID
```

For this configuration, the wrapper will loop over each initialization time and attempt to process all files that match that time. The wildcard character '*' will replace valid and lead in the template when attempting to

find data to process.

Runtime: Init: 2020-10-17 12Z Template Used: I2020101712_F*_V* Files Processed:

```
I2020101712_F003_V15
I2020101712_F006_V18
```

Runtime: Init: 2020-10-18 12Z Template Used: I2020101812_F*_V* Files Processed:

```
I2020101812_F003_V15
I2020101812_F006_V18
```

Runtime: Init: 2020-10-19 12Z Template Used: I2020101912_F*_V* Files Processed:

```
I2020101912_F003_V15
I2020101912_F006_V18
```

Note: If LOOP_BY was set to VALID, then the values defined by VALID_BEG, VALID_END, and VALID_INCREMENT would be substituted for the valid time while init and lead would be wildcard values.

Example 3 Run Once Per Forecast Lead Time:

```
[config]
SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_LEAD
```

For this configuration, the wrapper will loop over each forecast lead time and attempt to process all files that match that time. The wildcard character '*' will replace valid and init in the template when attempting to find data to process.

Runtime: Lead: 3 hour Template Used: I*_F003*_V* Files Processed:

```
I2020101712_F003_V15
I2020101812_F003_V15
I2020101912_F003_V15
```

Runtime: Lead: 6 hour Template Used: I*_F006*_V* Files Processed:

```
I2020101712_F006_V18
I2020101812_F006_V18
I2020101912_F006_V18
```

Example 4 Run Once For Each Time:

```
[config]
SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_FOR_EACH
```

For this configuration, the wrapper will loop over each initialization time and forecast lead times and attempt to process all files that match that time. The wildcard character '*' will replace valid only in the template when attempting to find data to process.

Runtime: Init: 2020-10-17 12Z, Forecast: 3 hour Template Used: I2020101712_F003_V* Files Processed:

```
I2020101712_F003_V15
```

Runtime: Init: 2020-10-17 12Z, Forecast: 6 hour Template Used: I2020101712_F006_V* Files Processed:

```
I2020101712_F006_V18
```

Runtime: Init: 2020-10-18 12Z, Forecast: 3 hour Template Used: I2020101812_F003_V* Files Processed:

```
I2020101812_F003_V15
```

Runtime: Init: 2020-10-18 12Z, Forecast: 6 hour Template Used: I2020101812_F006_V* Files Processed:

```
I2020101812_F006_V18
```

Runtime: Init: 2020-10-19 12Z, Forecast: 3 hour Template Used: I2020101912_F003_V* Files Processed:

```
I2020101912_F003_V15
```

Runtime: Init: 2020-10-19 12Z, Forecast: 6 hour Template Used: I2020101912_F006_V* Files Processed:

```
I2020101912_F006_V18
```

5.5.8 Config Utilities

5.5.8.1 Begin End Increment (begin_end_incr)

In configuration variables that can accept a list of values, the Begin End Increment utility can be used to easily create a sequence of numbers without having to type out the entire list explicitly. This functionality is similar to the Python range() function except that it is inclusive, meaning that the end value is also included in the list.

The notation is **begin_end_incr(b,e,i)** where b = the first lead value, e = the last lead value (inclusive), and i = the increment between values.

begin_end_incr(0,6,2) will expand to a list of numbers from 0 to 6 by 2, or 0, 2, 4, 6.

An optional 4th argument can be provided to specify the zero padding. begin_end_incr(8,10,1,2) will expand to 08, 09, 10.

If this syntax is found within a configuration variable, it will expand a string into a list with each number included. For example:

```
INPUT_TEMPLATE = ens01.nc, ens02.nc, ens03.nc, ens04.nc, ens05.nc, ens06.nc, ens07.nc, ens08.
↪nc
```

can be simplified as:

```
INPUT_TEMPLATE = ensbegin_end_incr(1,8,1,2).nc
```

5.6 How METplus controls MET configuration variables

METplus provides powerful user control of MET tool configuration file settings. If a MET tool uses a configuration file, then the corresponding METplus wrapper supports METplus configuration variables that control the MET tool configuration file settings. **The METplus wrappers provide a special "wrapped" MET configuration file that references environment variables that are set by the wrappers based on the values set in the METplus configuration files. THE USER SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!**

If there is a setting in the MET configuration file that is not currently supported by METplus that the user would like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87).

The following section demonstrates a few examples using GridStat.

5.6.1 GridStat Simple Example

Visit the *GridStat MET Configuration* (page 152) section of the User's Guide. This section contains a link to the default GridStat MET config file, which is found locally in **share/met/config/GridStatConfig_default** under the *MET_INSTALL_DIR* (page 38). Next the content of the wrapped GridStat configuration file (**parm/met_config/GridStatConfig_wrapped**) is displayed. Notice that this file is similar to the default GridStat MET config file, but some of the variables in the wrapped configuration file have been replaced with environment variables.

GridStatConfig_**default**:

```
desc = "NA";
```

GridStatConfig_**wrapped**:

```
// desc =
${METPLUS_DESC}
```

When GridStat is run, the tool first reads its default configuration file (GridStatConfig_**default**) and sets all of the default values. Then it reads the configuration file that is passed into the tool on the command line, which is *typically* the wrapped GridStat config file (parm/met_config/GridStatConfig_**wrapped**).

If the user sets the following in their METplus config file:

```
GRID_STAT_DESC = my_description
```

METplus will set the value of the ${METPLUS_DESC} environment variable to:

```
desc = "my_description";
```

Notice that the variable name and equals sign is included in the value of the environment variable. The default value for *desc* will be replaced with the new value "my_description" when the wrapped config file is read.

If the user does not set *GRID_STAT_DESC* in their METplus config files, then METplus will set the value of the ${METPLUS_DESC} environment variable to an empty string. This will result in the default value "NA" to be used.

Typically for single value or array MET config variables, the names of the METplus config variable, environment variable, and MET config variable are closely related, i.e.

- **desc**: MET config name
- GRID_STAT_**DESC**: METplus config name
- $METPLUS_**DESC**: Environment variable name

However, this is not always the case. Refer to the 'MET Configuration' section for each wrapper in the:doc:*wrappers* chapter to see the full list of supported variables.

5.6.2 GridStat Dictionary example

The MET configuration files may contain dictionaries that contain multiple variables within a variable. For example:

```
regrid = {
   to_grid    = NONE;
   method     = NEAREST;
   width      = 1;
   vld_thresh = 0.5;
   shape      = SQUARE;
}
```

The *regrid* dictionary contains 5 variables named *to_grid*, *method*, *width*, *vld_thresh*, and *shape*.

If only one or a few of the dictionary items are supported through the METplus wrappers, then they are handled in the same way as single value or array values described above. However, if the entire dictionary is supported, then it must be handled a little differently. The reason is MET will throw an error if it encounters a dictionary with no values inside, like this:

```
regrid = {}
```

To handle this, the values for the entire dictionary are handled in a single environment variable with a name that ends with "_DICT" to signify that it sets values for a dictionary:

```
// regrid = {
${METPLUS_REGRID_DICT}
```

Notice that the naming convention is still similar to the name of the MET config variable name.

Instead of a single METplus configuration variable to control the value of this environment variable, there are multiple variables – one for each item of the dictionary:

- GRID_STAT_REGRID_**TO_GRID**
- GRID_STAT_REGRID_**METHOD**
- GRID_STAT_REGRID_**WIDTH**
- GRID_STAT_REGRID_**VLD_THRESH**
- GRID_STAT_REGRID_**SHAPE**

If all of these variables are unset, then the value of ${METPLUS_REGRID_DICT} will be an empty string. If one or more of these variables are set, then each item will be formatted and added to the regrid dictionary.

If the following variable is set:

```
GRID_STAT_REGRID_TO_GRID = OBS
```

then ${METPLUS_REGRID_DICT} will be set to:

```
regrid = {to_grid = OBS;}
```

If the following variables are set:

```
GRID_STAT_REGRID_TO_GRID = OBS
GRID_STAT_REGRID_WIDTH = 2
```

then ${METPLUS_REGRID_DICT} will be set to:

```
regrid = {to_grid = OBS; width = 2;}
```

When a subset of a dictionary is defined in a MET configuration file, only the variables that are re-defined are replaced. The other dictionary items that are absent will use the default value.

5.6.3 GridStat Fields

Field information, i.e. the fcst/obs dictionary field item, is handled a little differently than other MET variables. Multiple fields can be specified for a given use case to generate a command for each field or, if the MET tool supports it, pass in all of the fields to a single command. Refer to the *Field Info* (page 58) section for information on how to sets these values.

5.7 Reconcile Default Values

While adding support for setting many new MET configuration variables through METplus wrapper configuration variables, it was discovered that some of the values set in the wrapped MET config files (found in *parm/met_config*) were different than the MET default values (found in *MET_INSTALL_DIR* (page 38)/share/met/config). Starting in v4.0.0, when a METplus configuration variable that overrides a MET variable is not set, the default MET value is used. Due to the disconnect between the wrapped config values and default values, some of the default settings will now differ if the wrapped MET configuration file found in *parm/met_config* is used in a use case. For more information regarding this logic, see the *How METplus controls MET configuration variables* (page 74) section.

This section lists all of the default values that have changed in the wrapped MET configuration files and the corresponding METplus configuration key/value pair to use to set the values to the previous default value. Note that any dictionary variables listed only include the variables inside that have changed, not the full set of variables that the dictionary contains.

5.7.1 EnsembleStatConfig

5.7.1.1 message_type

Old (Incorrect):	message_type = ["ADPSFC"];
New (Correct):	message_type = ["ADPUPA"];
METplus Config:	*ENSEMBLE_STAT_MESSAGE_TYPE* = ADPSFC

5.7.1.2 climo_cdf.cdf_bins

Old (Incorrect):	climo_cdf = {
	cdf_bins = 1;
	}

New (Correct):	climo_cdf = {
	cdf_bins = 10;
	}

METplus Config:	*ENSEMBLE_STAT_CLIMO_CDF_BINS* = 1

5.7.1.3 mask.poly

Old (Incorrect):	mask = {
	poly = [
	"MET_BASE/poly/HMT_masks/huc4_1605_poly.nc",
	"MET_BASE/poly/HMT_masks/huc4_1803_poly.nc"
	"MET_BASE/poly/HMT_masks/huc4_1804_poly.nc"
	"MET_BASE/poly/HMT_masks/huc4_1805_poly.nc"
	"MET_BASE/poly/HMT_masks/huc4_1806_poly.nc"
	];
	}

New (Correct):	mask = {
	poly = [];
	}

METplus Config:	*ENSEMBLE_STAT_MASK_POLY* = MET_BASE/poly/HMT_masks/huc4_1605_poly.nc, MET_BASE/poly/HMT_masks/huc4_1803_poly.nc, MET_BASE/poly/HMT_masks/huc4_1804_poly.nc, MET_BASE/poly/HMT_masks/huc4_1805_poly.nc, MET_BASE/poly/HMT_masks/huc4_1806_poly.nc

5.7.1.4 output_flag (multiple items)

Old (Incorrect):	output_flag = {
	ecnt = BOTH;
	rhist = BOTH;
	phist = BOTH;
	orank = BOTH;
	ssvar = BOTH;
	relp = BOTH;
	}

New (Correct):	output_flag = {
	ecnt = NONE;
	rps = NONE;
	rhist = NONE;
	phist = NONE;
	orank = NONE;
	ssvar = NONE;
	relp = NONE;
	}

METplus Config:	*ENSEMBLE_STAT_OUTPUT_FLAG_ECNT* = BOTH *ENSEMBLE_STAT_OUTPUT_FLAG_RHIST* = BOTH *ENSEMBLE_STAT_OUTPUT_FLAG_PHIST* = BOTH *ENSEMBLE_STAT_OUTPUT_FLAG_ORANK* = BOTH *ENSEMBLE_STAT_OUTPUT_FLAG_SSVAR* = BOTH *ENSEMBLE_STAT_OUTPUT_FLAG_RELP* = BOTH

5.7.2 GridStatConfig

5.7.2.1 cat_thresh

Old (Incorrect):	cat_thresh = [NA];
New (Correct):	cat_thresh = [];
METplus Config:	*GRID_STAT_MET_CONFIG_OVERRIDES* = cat_thresh = [NA];

5.7.2.2 output_flag (multiple items)

Old (Incorrect):	output_flag = {
	ctc = STAT;
	cts = STAT;
	grad = BOTH;
	}

New (Correct):	output_flag = {
	ctc = NONE;
	cts = NONE;
	grad = NONE;
	}

METplus Config:	*GRID_STAT_OUTPUT_FLAG_CTC* = STAT *GRID_STAT_OUTPUT_FLAG_CTS* = STAT *GRID_STAT_OUTPUT_FLAG_GRAD* = BOTH

5.7.2.3 nc_pairs_flag (multiple items)

Old (Incorrect):	nc_pairs_flag = {
	latlon = FALSE;
	raw = FALSE;
	diff = FALSE;
	climo = FALSE;
	apply_mask = FALSE;
	}

New (Correct):	nc_pairs_flag = {
	latlon = TRUE;
	raw = TRUE;
	diff = TRUE;
	climo = TRUE;
	apply_mask = TRUE;
	}

METplus Config:	*GRID_STAT_NC_PAIRS_FLAG_LATLON* = FALSE *GRID_STAT_NC_PAIRS_FLAG_RAW* = FALSE *GRID_STAT_NC_PAIRS_FLAG_DIFF* = FALSE *GRID_STAT_NC_PAIRS_FLAG_CLIMO* = FALSE *GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK* = FALSE

5.7.3 MODEConfig

5.7.3.1 grid_res

Old (Incorrect):	grid_res = 40;
New (Correct):	grid_res = 4;
METplus Config:	*MODE_GRID_RES* = 40

5.7.3.2 fcst.merge_thresh and fcst.merge_flag

Old (Incorrect):	fcst = {
	merge_thresh = >=75.0;
	merge_flag = NONE;
	}

New (Correct):	fcst = {
	merge_thresh = >=1.25;
	merge_flag = THRESH;
	}

METplus Config:	*MODE_FCST_MERGE_THRESH* = >=75.0 *MODE_FCST_MERGE_FLAG* = NONE *MODE_OBS_MERGE_THRESH* = >=75.0 *MODE_OBS_MERGE_FLAG* = NONE

5.7.3.3 fcst_raw_plot.color_table

Old (Incorrect):	fcst_raw_plot = {
	color_table =
	"MET_BASE/colortables/mode_raw.ctable";
	}

New (Correct):	fcst_raw_plot = {
	color_table =
	"MET_BASE/colortables/met_default.ctable";
	}

METplus Config:	*MODE_MET_CONFIG_OVERRIDES* = fcst_raw_plot = {color_table = "MET_BASE/colortables/mode_raw.ctable";}

5.7.3.4 obs_raw_plot.color_table

Old (Incorrect):	obs_raw_plot = {
	color_table = "MET_BASE/colortables/mode_raw.ctable";
	}

New (Correct):	obs_raw_plot = {
	color_table = "MET_BASE/colortables/met_default.ctable";
	}

METplus Config:	*MODE_MET_CONFIG_OVERRIDES* = obs_raw_plot = {color_table = "MET_BASE/colortables/mode_raw.ctable";}

5.7.3.5 mask_missing_flag

Old (Incorrect):	mask_missing_flag = BOTH;
New (Correct):	mask_missing_flag = NONE;
METplus Config:	*MODE_MASK_MISSING_FLAG* = BOTH

5.7.4 PB2NCConfig

5.7.4.1 level_category

Old (Incorrect):	level_category = [0, 1, 4, 5, 6];
New (Correct):	level_category = [];
METplus Config:	*PB2NC_LEVEL_CATEGORY* = 0, 1, 4, 5, 6

5.7.4.2 quality_mark_thresh

Old (Incorrect):	quality_mark_thresh = 3;
New (Correct):	quality_mark_thresh = 2;
METplus Config:	*PB2NC_QUALITY_MARK_THRESH* = 3

5.7.4.3 time_summary.step and time_summary.width

Old (Incorrect):	time_summary = {
	step = 3600;
	width = 3600;
	}

New (Correct):	time_summary = {
	step = 300;
	width = 600;
	}

METplus Config:	*PB2NC_TIME_SUMMARY_STEP* = 3600 *PB2NC_TIME_SUMMARY_WIDTH* = 3600

5.7.4.4 pb_report_type

Old (Incorrect):	pb_report_type = [120, 220, 221, 122, 222, 223, 224, 133, 233, 188, 288, 180, 280, 181, 182, 281, 282, 183, 284, 187, 287];
New (Correct):	pb_report_type = [];
METplus Config:	*PB2NC_PB_REPORT_TYPE* = 120, 220, 221, 122, 222, 223, 224, 133, 233, 188, 288, 180, 280, 181, 182, 281, 282, 183, 284, 187, 287

5.7.5 PointStatConfig

5.7.5.1 regrid.method and regrid_width

Old (Incorrect):	regrid = {
	method = BILIN;
	width = 2;
	}

New (Correct):	regrid = {
	method = NEAREST;
	width = 1;
	}

METplus Config:	*POINT_STAT_REGRID_METHOD* = BILIN *POINT_STAT_REGRID_WIDTH* = 2

5.7.5.2 obs_quality

Old (Incorrect):	obs_quality = ["1", "2", "3"];
New (Correct):	obs_quality = [];
METplus Config:	*POINT_STAT_OBS_QUALITY* = 1, 2, 3

5.7.5.3 climo_mean.time_interp_method and climo_stdev.time_interp_method

Old (Incorrect):	climo_mean = {
	time_interp_method = NEAREST;
	}
	climo_stdev = {
	time_interp_method = NEAREST;
	}

New (Correct):	climo_mean = {
	time_interp_method = DW_MEAN;
	}
	climo_stdev = {
	time_interp_method = DW_MEAN;
	}

METplus Config:	*POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD* = NEAREST *POINT_STAT_CLIMO_STDEV_TIME_INTERP_METHOD* = NEAREST

5.7.5.4 interp.type.method and interp.type.width

Old (Incorrect):	interp = {
	type = [
	{
	method = BILIN;
	width = 2;
	}
	];
	}

New (Correct):	interp = {
	type = [
	{
	method = NEAREST;
	width = 1;
	}
	];
	}

METplus Config:	*POINT_STAT_INTERP_TYPE_METHOD* = BILIN *POINT_STAT_INTERP_TYPE_WIDTH* = 2

5.8 Overriding Unsupported MET configuration variables

While METplus does provide support for overriding many of the commonly used MET config variables through the wrappers, there will certainly be instances when a user wishes to control a MET config variable that is not supported in the METplus configuration. Wrappers for MET tools that utilize configuration files support a METplus configuration variable used to override any unsupported MET config variables. These variables contain the name of the MET tool (in all caps) followed by _MET_CONFIG_OVERRIDES. Here are some examples:

- *ENSEMBLE_STAT_MET_CONFIG_OVERRIDES*
- *ASCII2NC_MET_CONFIG_OVERRIDES*
- *GRID_DIAG_MET_CONFIG_OVERRIDES*
- *GRID_STAT_MET_CONFIG_OVERRIDES*

- *MODE_MET_CONFIG_OVERRIDES*
- *MTD_MET_CONFIG_OVERRIDES*
- *PB2NC_MET_CONFIG_OVERRIDES*
- *POINT_STAT_MET_CONFIG_OVERRIDES*
- *SERIES_ANALYSIS_MET_CONFIG_OVERRIDES*
- *STAT_ANALYSIS_MET_CONFIG_OVERRIDES*
- *TC_GEN_MET_CONFIG_OVERRIDES*
- *TC_PAIRS_MET_CONFIG_OVERRIDES*
- *TC_RMW_MET_CONFIG_OVERRIDES*
- *TC_STAT_MET_CONFIG_OVERRIDES*

The value set for each of these variables are set to the **${METPLUS_MET_CONFIG_OVERRIDES}** environment variable for the corresponding MET tool. This environment variable is referenced at the bottom of each wrapped MET configuration file, so the values are read at the end of of parsing, overriding any values that were set.

Note: We recommend using this approach to controlling unsupported MET config options over using a modified MET configuration file, although this approach is still supported. Newly added features and variable override support may be more difficult to incorporate using the latter approach. Please create a post in the METplus GitHub Discussions Forum for assistance with updating a use case to migrate away from using a modified MET configuration file.

5.8.1 MET Config Override GridStat Simple Example

Let's use the example of a user running GridStat. The user has a customized GridStat verification task, and needs a specialized setting in the 'distance_map' dictionary in the MET GridStat configuration file. Here's what the default MET config file looks like:

```
distance_map = {
 baddeley_p        = 2;
 baddeley_max_dist = NA;
 fom_alpha         = 0.1;
 zhu_weight        = 0.5;
}
```

Currently there is no support in METplus to control any of these items specifically, however they can be set using *GRID_STAT_MET_CONFIG_OVERRIDES*. Recall from *How METplus controls MET configuration variables* (page 74) that METplus will utilize the default settings for each variable in the 'distance_map' dictionary. If a user wishes to override the default value of the 'baddeley_p' variable, then they would create the following entry in their METplus configuration file:

```
GRID_STAT_MET_CONFIG_OVERRIDES = distance_map = {baddeley_p = 10;}
```

This is quite confusing to read since there are three '=' characters, however METplus interprets everything to the right of the first '=' character (reading left –> right) as a single string. In this case the value is **'distance_map = {baddeley_p = 10;}'**. When METplus runs GridStat, it appends the 'distance_map' dictionary to the end of the wrapped GridStat MET configuration file to override the default value of the 'baddeley_p' variable in the 'distance_map' dictionary. A line would be added that looks like:

```
distance_map = {baddeley_p = 10;}
```

This causes MET to update the value of the 'baddeley_p' variable in the 'distance_map' dictionary to be 10 instead of the default value of 2.

More than one MET config variables can be set using this functionality. Simply list all of the overrides in the same METplus configuration variable:

```
GRID_STAT_MET_CONFIG_OVERRIDES = distance_map = {baddeley_p = 10;} rank_corr_flag = TRUE;
```

The values must match the format of the variables in the default MET configuration file with a semi-colon after single values and arrays and curly braces around dictionaries.

5.9 User Environment Variables

In addition to the environment variables that the METplus wrappers set automatically before running applications, users can define additional environment variables. These environment variables will only be set in the environment that runs the commands, so the user's environment is preserved.

This capability is useful when calling a script (such as a UserScript command or a Python embedding script) that requires many inputs from the user. Instead of calling the script and passing in all of the values as command line arguments, the environment variables can be read from inside the script.

To set a user-defined environment variable, add a section to a METplus configuration files called [user_env_vars]. Under this header, add key-value pairs as desired. For example, if the following is added to a METplus configuration file:

```
[user_env_vars]
VAR_NAME = some_text_for_feb_1_1987_run
```

then an environment variable named "VAR_NAME" set to the value "some_text_for_feb_1_1987_run" will be set in the environment for every command run by the METplus wrappers.

This is the equivalent of running this bash command:

```
$ export VAR_NAME=some_text_for_feb_1_1987_run
```

on the command line before calling run_metplus.py.

Other variables can also be referenced in the METplus config file. For example:

```
[config]
INIT_BEG = 1987020104

[user_env_vars]
USE_CASE_TIME_ID = {INIT_BEG}
```

This is the equivalent of running this bash command:

```
$ export USE_CASE_TIME_ID=1987020104
```

on the command line before calling run_metplus.py.

Note: In previous versions of METplus, we recommended using this to control unsupported MET config file options. Since this requires also modifying the MET config file used by METplus, we no longer recommend this. Instead, we strongly encourage the user to use the new capability defined in *Overriding Unsupported MET configuration variables* (page 87).

5.10 Setting Config Variables with Environment Variables

The METplus config variables can be set to the value of local environment variables when METplus is run. To set any METplus config variable to the value of a local environment variable, use the following syntax:

```
METPLUS_MY_VAR = {ENV[LOCAL_ENV_VAR]}
```

If the following bash command is run before calling run_metplus.py:

```
export LOCAL_ENV_VAR=my_value
```

then the METplus configuration variable METPLUS_MY_VAR will be set to my_value.

5.11 Updating Configuration Files - Handling Deprecated Configuration Variables

If upgrading from a METplus version earlier than v3.0, this content is important to getting started using a newly released version. **If upgrading from METplus v3.0 and above or if installing METplus for the first time, please skip this section.**

METplus developers strive to allow backwards compatibility so new versions of the tools will continue to work as they did in previous versions. However, sometimes changes are necessary for clarity and cohesion. Many configuration variable names have changed in version 3.0 in an attempt to make their function more clear. If any deprecated METplus configuration variables are found in a user's use case, execution will stop immediately and an error report of all variables that must be updated is output. In some cases, simply renaming the variable is sufficient. Other changes may require more thought. The next few sections will outline a few of common changes that will need to be made. In the last section, a tool called validate_config.py

is described. This tool can be used to help with this transition by automating some of the work required to update configuration files.

5.11.1 Simple Rename

In most cases, there is a simple one-to-one relationship between a deprecated configuration variable and a valid one. In this case, renaming the variable will resolve the issue.

Example:

```
ERROR: DEPRECATED CONFIG ITEMS WERE FOUND. PLEASE REMOVE/REPLACE THEM FROM CONFIG FILES
ERROR: [dir] MODEL_DATA_DIR should be replaced with EXTRACT_TILES_GRID_INPUT_DIR
ERROR: [config] STAT_LIST should be replaced with SERIES_ANALYSIS_STAT_LIST
```

These cases can be handled automatically by using the *Validate Config Helper Script* (page 94).

5.11.2 FCST/OBS/BOTH Variables

Field information passed into many of the MET tools is defined with the [FCST/OBS]_VAR<n>_[NAME/LEVELS/THRESH/OPTIONS] configuration variables. For example, FCST_VAR1_NAME and FCST_VAR1_LEVELS are used to define forecast name/level values that are compared to observations defined with OBS_VAR1_NAME and OBS_VAR1_LEVELS.

Before METplus 3.0, users could define the FCST_* variables and omit the OBS_* variables or vice versa. In this case, it was assumed the undefined values matched the coresponding term. For example, if FCST_VAR1_NAME = TMP and OBS_VAR1_NAME is not defined, it was assumed that OBS_VAR1_NAME = TMP as well. This method was not always clear to users.

Starting in METplus 3.0, users are required to either explicitly set both FCST_* and OBS_* variables or set the equivalent BOTH_* variables to make it clear that the values apply to both forecast and observation data.

Example:

```
ERROR: If FCST_VAR1_NAME is set, the user must either set OBS_VAR1_NAME or change FCST_VAR1_
↪NAME to BOTH_VAR1_NAME
ERROR: If FCST_VAR2_NAME is set, the user must either set OBS_VAR2_NAME or change FCST_VAR2_
↪NAME to BOTH_VAR2_NAME
ERROR: If FCST_VAR1_LEVELS is set, the user must either set OBS_VAR1_LEVELS or change FCST_
↪VAR1_LEVELS to BOTH_VAR1_LEVELS
ERROR: If FCST_VAR2_LEVELS is set, the user must either set OBS_VAR2_LEVELS or change FCST_
↪VAR2_LEVELS to BOTH_VAR2_LEVELS
```

These cases can be handled automatically by using the *Validate Config Helper Script* (page 94), but users should review the suggested changes, as they may want to update differently.

5.11.3 PCPCombine Input Levels

Prior to METplus 3.0, the PCPCombine wrapper only allowed the user to define a single input accumulation amount to be used to build a desired accumulation. However, some data sets include more than one accumulation field. PCPCombine wrapper was enhanced in version 3.0 to allow users to specify a list of accumulations available in the input data. Instead of only being able to specify FCST_PCP_COMBINE_INPUT_LEVEL, users can now specify a list of accumulations with *FCST_PCP_COMBINE_INPUT_ACCUMS*.

Example:

```
ERROR: [config] OBS_PCP_COMBINE_INPUT_LEVEL should be replaced with OBS_PCP_COMBINE_INPUT_
↪ACCUMS
```

These cases can be handled automatically by using the *Validate Config Helper Script* (page 94), but users should review the suggested changes, as they may want to include other available input accumulations.

5.11.4 MET Configuration Files

The METplus wrappers set environment variables that are read by the MET configuration files to customize each run. Some of the environment variables that were previously set by METplus wrappers to handle very specific use cases are no longer set in favor of using a common set of variables across the MET tools. The following are examples of changes that have occurred in METplus regarding environment variables.

EnsembleStat previously set $GRID_VX to define the grid to use to regrid data within the tool. In version 3.0, MET tools that have a 'to_grid' value in the 'grid' dictionary of the MET config file have a uniformly named METplus configuration variable called <MET-tool>_REGRID_TO_GRID (i.e. *ENSEMBLE_STAT_REGRID_TO_GRID*) that is used to define this value:

```
Before:
   to_grid    = ${GRID_VX};

After:
   to_grid    = ${REGRID_TO_GRID};
```

MET_VALID_HHMM was used by GridStat wrapper to set part of the climatology file path. This was replaced by the METplus configuration variables <MET-tool>_CLIMO_[MEAN/STDEV]_INPUT_[DIR/TEMPLATE] (i.e. *GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE*):

```
Before:
   file_name = [ "${INPUT_BASE}/grid_to_grid/nwprod/fix/cmean_1d.1959${MET_VALID_HHMM}" ];

After:
   file_name = [ ${CLIMO_MEAN_FILE} ];
```

The output_prefix variable in the MET config files was previously set by referencing variable environment variables set by METplus. This has since been changed so that output_prefix references the $OUTPUT_PREFIX environment variable. This value is now set in the METplus configuration files using the wrapper-specific configuration variable, such as *GRID_STAT_OUTPUT_PREFIX* or *ENSEMBLE_STAT_OUTPUT_PREFIX*:

```
Before:
   output_prefix    = "${FCST_VAR}_vs_${OBS_VAR}";

After:
   output_prefix    = "${OUTPUT_PREFIX}";
```

Due to these changes, MET configuration files that refer to any of these deprecated environment variables will throw an error. While the *Validate Config Helper Script* (page 94) will automatically remove any invalid environment variables that may be set in the MET configuration files, the user will be responsible for adding the corresponding METplus configuration variable to reproduce the intended behavior. The tool will give a suggested value for <MET-tool>_OUTPUT_PREFIX.

Example log output:

```
DEBUG: Checking for deprecated environment variables in: DeprecatedConfig
ERROR: Please remove deprecated environment variable ${GRID_VX} found in MET config file:␣
↪DeprecatedConfig
ERROR: MET to_grid variable should reference ${REGRID_TO_GRID} environment variable
INFO: Be sure to set GRID_STAT_REGRID_TO_GRID to the correct value.

ERROR: Please remove deprecated environment variable ${MET_VALID_HHMM} found in MET config␣
↪file: DeprecatedConfig
ERROR: Set GRID_STAT_CLIMO_MEAN_INPUT_[DIR/TEMPLATE] in a METplus config file to set CLIMO_
↪MEAN_FILE in a MET config

ERROR: output_prefix variable should reference ${OUTPUT_PREFIX} environment variable
INFO: GRID_STAT_OUTPUT_PREFIX will need to be added to the METplus config file that sets␣
↪GRID_STAT_CONFIG_FILE. Set it to:
INFO: GRID_STAT_OUTPUT_PREFIX = {CURRENT_FCST_NAME}_vs_{CURRENT_OBS_NAME}
```

These cases can be handled automatically by using the *Validate Config Helper Script* (page 94), but users should review the suggested changes and make sure they add the appropriate recommended METplus configuration variables to their files to achieve the same behavior.

5.11.5 SED Commands

Running run_metplus.py with one or more configuration files that contain deprecated variables that can be fixed with a find/replace command will generate a file in the {OUTPUT_BASE} called sed_commands.txt. This file contains a list of commands that can be run to update the configuration file. Lines that start with "#Add" are intended to notify the user to add a variable to their METplus configuration file.

The *Validate Config Helper Script* (page 94) will step through each of these commands and execute them upon approval.

Example sed_commands.txt content:

```
sed -i 's|^   to_grid    = ${GRID_VX};|   to_grid    = ${REGRID_TO_GRID};|g' DeprecatedConfig
#Add GRID_STAT_REGRID_TO_GRID
```

(continues on next page)

(continued from previous page)

```
sed -i 's|^   file_name = [ "${INPUT_BASE}/grid_to_grid/nwprod/fix/cmean_1d.1959${MET_VALID_
↪HHMM}" ];|   file_name = [ ${CLIMO_MEAN_FILE} ];|g' DeprecatedConfig
#Add GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE
sed -i 's|^output_prefix    = "${FCST_VAR}_vs_${OBS_VAR}";|output_prefix    = "${OUTPUT_
↪PREFIX}";|g' DeprecatedConfig
#Add GRID_STAT_OUTPUT_PREFIX = {CURRENT_FCST_NAME}_vs_{CURRENT_OBS_NAME}
sed -i 's|^FCST_VAR1_NAME|BOTH_VAR1_NAME|g' deprecated.conf
sed -i 's|^FCST_VAR1_LEVELS|BOTH_VAR1_LEVELS|g' deprecated.conf
```

5.11.6 Validate Config Helper Script

The script named validate_config.py is found in the same directory as run_metplus.py. To use this script, call it with the same arguments as run_metplus.py:

```
run_metplus.py  ./my_conf.py ./another_config.py
validate_config.py ./my_conf.py ./another_config.py
```

A valid configuration must be passed to the script, as in the user must properly set *MET_INSTALL_DIR*, *INPUT_BASE*, and *OUTPUT_BASE*, or it will not run.

The script will evaluate all of the configuration files, including any MET configuration file that is referenced in a _CONFIG_FILE variable, such as *GRID_STAT_CONFIG_FILE*. For each deprecated item that is found, the script will suggest a replacement for the file where the deprecated item was found.

Example 1 (Simple Rename):

```
The following replacement is suggested for ./deprecated.conf

Before:
STAT_LIST = TOTAL, OBAR, FBAR

After:
SERIES_ANALYSIS_STAT_LIST = TOTAL, OBAR, FBAR

Would you like the make this change to ./deprecated.conf? (y/n)[n]
```

Example 2 (FCST/OBS/BOTH Variables):

```
The following replacement is suggested for ./deprecated.conf

Before:
FCST_VAR1_NAME = TMP

After:
BOTH_VAR1_NAME = TMP
```

(continues on next page)

(continued from previous page)

```
Would you like the make this change to ./deprecated.conf? (y/n)[n]
```

Example 3 (PCPCombine Input Levels):

```
The following replacement is suggested for ./deprecated.conf

Before:
OBS_PCP_COMBINE_INPUT_LEVEL = 6

After:
OBS_PCP_COMBINE_INPUT_ACCUMS = 6

Would you like the make this change to ./deprecated.conf? (y/n)[n]
```

Example 4 (MET Configuration File):

```
The following replacement is suggested for DeprecatedConfig

Before:
   to_grid    = ${GRID_VX};

After:
   to_grid    = ${REGRID_TO_GRID};

Would you like the make this change to DeprecatedConfig? (y/n)[n]

IMPORTANT: If it is not already set, add the following in the [config] section to your␣
↪METplus configuration file that sets GRID_STAT_CONFIG_FILE:

GRID_STAT_REGRID_TO_GRID
Make this change before continuing! [OK]
```

Example 5 (Another MET Configuration File):

```
The following replacement is suggested for DeprecatedConfig

Before:
output_prefix    = "${FCST_VAR}_vs_${OBS_VAR}";

After:
output_prefix    = "${OUTPUT_PREFIX}";

Would you like the make this change to DeprecatedConfig? (y/n)[n]

IMPORTANT: If it is not already set, add the following in the [config] section to the␣
```

(continues on next page)

(continued from previous page)

```
↪METplus configuration file that sets GRID_STAT_CONFIG_FILE:

GRID_STAT_OUTPUT_PREFIX = {CURRENT_FCST_NAME}_vs_{CURRENT_OBS_NAME}
Make this change before continuing! [OK]
```

Note: While the METplus developers are very diligent to include deprecated variables in this functionality, some may slip through the cracks. When upgrading to a new version of METplus, it is important to test and review the use cases to ensure they produce the same results as the previous version. Please create a post in the METplus GitHub Discussions Forum with any questions.

Chapter 6

Python Wrappers

This chapter provides a description of each supported Python wrapper in METplus Wrappers. A wrapper is generally a Python script that encapsulates the behavior of a corresponding MET tool. Each of these sections can be added to the PROCESS_LIST configuration list variable. The METplus Configuration section of each wrapper section below lists the METplus Wrappers configuration variables that are specific to that wrapper organized by config file section. You can find more information about each item in the METplus Configuration Glossary. The MET Configuration section of each wrapper (if applicable) displays the wrapped MET configuration file that utilizes environment variables to override settings. These sections also contain a list of environment variables that are referenced in the wrapped MET configuration files and a table to show which METplus configuration variables are used to set them and which MET configuration variables they override.

6.1 ASCII2NC

6.1.1 Description

Used to configure the MET tool ASCII2NC

6.1.2 METplus Configuration

ASCII2NC_INPUT_DIR
ASCII2NC_OUTPUT_DIR
ASCII2NC_INPUT_TEMPLATE
ASCII2NC_OUTPUT_TEMPLATE
LOG_ASCII2NC_VERBOSITY
ASCII2NC_SKIP_IF_OUTPUT_EXISTS
ASCII2NC_CONFIG_FILE
ASCII2NC_FILE_WINDOW_BEGIN
ASCII2NC_FILE_WINDOW_END
ASCII2NC_WINDOW_BEGIN

ASCII2NC_WINDOW_END
ASCII2NC_INPUT_FORMAT
ASCII2NC_MASK_GRID
ASCII2NC_MASK_POLY
ASCII2NC_MASK_SID
ASCII2NC_TIME_SUMMARY_FLAG
ASCII2NC_TIME_SUMMARY_RAW_DATA
ASCII2NC_TIME_SUMMARY_BEG
ASCII2NC_TIME_SUMMARY_END
ASCII2NC_TIME_SUMMARY_STEP
ASCII2NC_TIME_SUMMARY_WIDTH
ASCII2NC_TIME_SUMMARY_GRIB_CODES
ASCII2NC_TIME_SUMMARY_VAR_NAMES
ASCII2NC_TIME_SUMMARY_TYPES
ASCII2NC_TIME_SUMMARY_VALID_FREQ
ASCII2NC_TIME_SUMMARY_VALID_THRESH
ASCII2NC_CUSTOM_LOOP_LIST
ASCII2NC_MET_CONFIG_OVERRIDES

6.1.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/Ascii2NcConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// Default ascii2nc configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to summarize the ASCII data read in
//

//
```

(continues on next page)

(continued from previous page)

```
// Time periods for the summarization
// obs_var (string array) is added and works like grib_code (int array)
// when the obs name is given instead of grib_code
//
${METPLUS_TIME_SUMMARY_DICT}

//
// Mapping of input little_r report types to output message types
//
message_type_map = [
   { key = "FM-12 SYNOP";  val = "ADPSFC"; },
   { key = "FM-13 SHIP";   val = "SFCSHP"; },
   { key = "FM-15 METAR";  val = "ADPSFC"; },
   { key = "FM-18 BUOY";   val = "SFCSHP"; },
   { key = "FM-281 QSCAT"; val = "ASCATW"; },
   { key = "FM-32 PILOT";  val = "ADPUPA"; },
   { key = "FM-35 TEMP";   val = "ADPUPA"; },
   { key = "FM-88 SATOB";  val = "SATWND"; },
   { key = "FM-97 ACARS";  val = "AIRCFT"; }
];

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_TIME_SUMMARY_DICT}

METplus Config(s)	MET Config File
ASCII2NC_TIME_SUMMARY_FLAG	time_summary.flag
ASCII2NC_TIME_SUMMARY_RAW_DATA	time_summary.raw_data
ASCII2NC_TIME_SUMMARY_BEG	time_summary.beg
ASCII2NC_TIME_SUMMARY_END	time_summary.end
ASCII2NC_TIME_SUMMARY_STEP	time_summary.step
ASCII2NC_TIME_SUMMARY_WIDTH	time_summary.width
ASCII2NC_TIME_SUMMARY_GRIB_CODES	time_summary.grib_code
ASCII2NC_TIME_SUMMARY_VAR_NAMES	time_summary.obs_var
ASCII2NC_TIME_SUMMARY_TYPES	time_summary.type
ASCII2NC_TIME_SUMMARY_VALID_FREQ	time_summary.vld_freq
ASCII2NC_TIME_SUMMARY_VALID_THRESH	time_summary.vld_thresh

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
ASCII2NC_MET_CONFIG_OVERRIDES	n/a

6.2 CyclonePlotter

6.2.1 Description

This wrapper does not have a corresponding MET tool but instead wraps the logic necessary to create plots of cyclone tracks. Currently only the output from the MET tc-pairs tool can be plotted. If used on an internet-limited system, additional dependencies may apply. See *Installation* (page 29) for details.

6.2.2 METplus Configuration

CYCLONE_PLOTTER_INPUT_DIR
CYCLONE_PLOTTER_OUTPUT_DIR
CYCLONE_PLOTTER_INIT_DATE
CYCLONE_PLOTTER_INIT_HR
CYCLONE_PLOTTER_MODEL
CYCLONE_PLOTTER_PLOT_TITLE
CYCLONE_PLOTTER_CIRCLE_MARKER_SIZE
CYCLONE_PLOTTER_CROSS_MARKER_SIZE
CYCLONE_PLOTTER_GENERATE_TRACK_ASCII
CYCLONE_PLOTTER_ADD_WATERMARK
CYCLONE_PLOTTER_GLOBAL_PLOT
CYCLONE_PLOTTER_WEST_LON
CYCLONE_PLOTTER_EAST_LON
CYCLONE_PLOTTER_NORTH_LAT
CYCLONE_PLOTTER_SOUTH_LAT
CYCLONE_PLOTTER_ANNOTATION_FONT_SIZE
CYCLONE_PLOTTER_RESOLUTION_DPI

Warning: DEPRECATED:

CYCLONE_OUT_DIR
CYCLONE_INIT_DATE

CYCLONE_INIT_HR
CYCLONE_MODEL
CYCLONE_PLOT_TITLE
CYCLONE_CIRCLE_MARKER_SIZE
CYCLONE_CROSS_MARKER_SIZE
CYCLONE_GENERATE_TRACK_ASCII

6.3 EnsembleStat

6.3.1 Description

Used to configure the MET tool ensemble_stat.

6.3.2 METplus Configuration

OBS_ENSEMBLE_STAT_POINT_INPUT_DIR
OBS_ENSEMBLE_STAT_GRID_INPUT_DIR
FCST_ENSEMBLE_STAT_INPUT_DIR
ENSEMBLE_STAT_OUTPUT_DIR
OBS_ENSEMBLE_STAT_POINT_INPUT_TEMPLATE
OBS_ENSEMBLE_STAT_GRID_INPUT_TEMPLATE
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE
FCST_ENSEMBLE_STAT_INPUT_FILE_LIST
ENSEMBLE_STAT_OUTPUT_TEMPLATE
ENSEMBLE_STAT_CTRL_INPUT_DIR
ENSEMBLE_STAT_CTRL_INPUT_TEMPLATE
ENSEMBLE_STAT_ENS_MEAN_INPUT_TEMPLATE
ENSEMBLE_STAT_ENS_MEAN_INPUT_DIR
LOG_ENSEMBLE_STAT_VERBOSITY
FCST_ENSEMBLE_STAT_INPUT_DATATYPE
OBS_ENSEMBLE_STAT_INPUT_POINT_DATATYPE
OBS_ENSEMBLE_STAT_INPUT_GRID_DATATYPE
ENSEMBLE_STAT_REGRID_TO_GRID
ENSEMBLE_STAT_REGRID_METHOD
ENSEMBLE_STAT_REGRID_WIDTH
ENSEMBLE_STAT_REGRID_VLD_THRESH
ENSEMBLE_STAT_REGRID_SHAPE
ENSEMBLE_STAT_REGRID_CONVERT
ENSEMBLE_STAT_REGRID_CENSOR_THRESH

ENSEMBLE_STAT_REGRID_CENSOR_VAL
ENSEMBLE_STAT_CONFIG_FILE
ENSEMBLE_STAT_MET_OBS_ERR_TABLE
ENSEMBLE_STAT_N_MEMBERS
OBS_ENSEMBLE_STAT_WINDOW_BEGIN
OBS_ENSEMBLE_STAT_WINDOW_END
OBS_ENSEMBLE_STAT_FILE_WINDOW_BEGIN
OBS_ENSEMBLE_STAT_FILE_WINDOW_END
ENSEMBLE_STAT_ENS_THRESH
ENSEMBLE_STAT_VLD_THRESH
ENSEMBLE_STAT_OBS_THRESH
ENSEMBLE_STAT_CUSTOM_LOOP_LIST
ENSEMBLE_STAT_SKIP_IF_OUTPUT_EXISTS
ENSEMBLE_STAT_DESC
ENSEMBLE_STAT_ENS_SSVAR_BIN_SIZE
ENSEMBLE_STAT_ENS_PHIST_BIN_SIZE
ENSEMBLE_STAT_CLIMO_CDF_BINS
ENSEMBLE_STAT_CLIMO_CDF_CENTER_BINS
ENSEMBLE_STAT_CLIMO_CDF_WRITE_BINS
ENSEMBLE_STAT_CLIMO_CDF_DIRECT_PROB
ENSEMBLE_STAT_DUPLICATE_FLAG
ENSEMBLE_STAT_SKIP_CONST
ENSEMBLE_STAT_CENSOR_THRESH
ENSEMBLE_STAT_CENSOR_VAL
ENSEMBLE_STAT_DUPLICATE_FLAG
ENSEMBLE_STAT_SKIP_CONST
ENSEMBLE_STAT_OBS_ERROR_FLAG
ENSEMBLE_STAT_CLIMO_MEAN_FILE_NAME
ENSEMBLE_STAT_CLIMO_MEAN_VAR<n>_NAME
ENSEMBLE_STAT_CLIMO_MEAN_VAR<n>_LEVELS
ENSEMBLE_STAT_CLIMO_MEAN_VAR<n>_OPTIONS
ENSEMBLE_STAT_CLIMO_MEAN_FIELD
ENSEMBLE_STAT_CLIMO_MEAN_REGRID_METHOD
ENSEMBLE_STAT_CLIMO_MEAN_REGRID_WIDTH
ENSEMBLE_STAT_CLIMO_MEAN_REGRID_VLD_THRESH
ENSEMBLE_STAT_CLIMO_MEAN_REGRID_SHAPE
ENSEMBLE_STAT_CLIMO_MEAN_TIME_INTERP_METHOD
ENSEMBLE_STAT_CLIMO_MEAN_MATCH_MONTH
ENSEMBLE_STAT_CLIMO_MEAN_DAY_INTERVAL
ENSEMBLE_STAT_CLIMO_MEAN_HOUR_INTERVAL
ENSEMBLE_STAT_CLIMO_MEAN_USE_FCST

ENSEMBLE_STAT_CLIMO_MEAN_USE_OBS
ENSEMBLE_STAT_CLIMO_STDEV_FILE_NAME
ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_NAME
ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_LEVELS
ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_OPTIONS
ENSEMBLE_STAT_CLIMO_STDEV_FIELD
ENSEMBLE_STAT_CLIMO_STDEV_REGRID_METHOD
ENSEMBLE_STAT_CLIMO_STDEV_REGRID_WIDTH
ENSEMBLE_STAT_CLIMO_STDEV_REGRID_VLD_THRESH
ENSEMBLE_STAT_CLIMO_STDEV_REGRID_SHAPE
ENSEMBLE_STAT_CLIMO_STDEV_TIME_INTERP_METHOD
ENSEMBLE_STAT_CLIMO_STDEV_MATCH_MONTH
ENSEMBLE_STAT_CLIMO_STDEV_DAY_INTERVAL
ENSEMBLE_STAT_CLIMO_STDEV_HOUR_INTERVAL
ENSEMBLE_STAT_CLIMO_STDEV_USE_FCST
ENSEMBLE_STAT_CLIMO_STDEV_USE_OBS
ENSEMBLE_STAT_MASK_GRID
ENSEMBLE_STAT_CI_ALPHA
ENSEMBLE_STAT_INTERP_FIELD
ENSEMBLE_STAT_INTERP_VLD_THRESH
ENSEMBLE_STAT_INTERP_SHAPE
ENSEMBLE_STAT_INTERP_METHOD
ENSEMBLE_STAT_INTERP_WIDTH
ENSEMBLE_STAT_OUTPUT_FLAG_ECNT
ENSEMBLE_STAT_OUTPUT_FLAG_RPS
ENSEMBLE_STAT_OUTPUT_FLAG_RHIST
ENSEMBLE_STAT_OUTPUT_FLAG_PHIST
ENSEMBLE_STAT_OUTPUT_FLAG_ORANK
ENSEMBLE_STAT_OUTPUT_FLAG_SSVAR
ENSEMBLE_STAT_OUTPUT_FLAG_RELP
ENSEMBLE_STAT_OUTPUT_FLAG_PCT
ENSEMBLE_STAT_OUTPUT_FLAG_PSTD
ENSEMBLE_STAT_OUTPUT_FLAG_PJC
ENSEMBLE_STAT_OUTPUT_FLAG_PRC
ENSEMBLE_STAT_OUTPUT_FLAG_ECLV
ENSEMBLE_STAT_NC_ORANK_FLAG_LATLON
ENSEMBLE_STAT_NC_ORANK_FLAG_MEAN
ENSEMBLE_STAT_NC_ORANK_FLAG_RAW
ENSEMBLE_STAT_NC_ORANK_FLAG_RANK
ENSEMBLE_STAT_NC_ORANK_FLAG_PIT
ENSEMBLE_STAT_NC_ORANK_FLAG_VLD_COUNT

ENSEMBLE_STAT_NC_ORANK_FLAG_WEIGHT
ENSEMBLE_STAT_OBS_QUALITY_INC
ENSEMBLE_STAT_OBS_QUALITY_EXC
ENSEMBLE_STAT_MET_CONFIG_OVERRIDES
ENSEMBLE_STAT_ENS_MEMBER_IDS
ENSEMBLE_STAT_CONTROL_ID
ENSEMBLE_STAT_GRID_WEIGHT_FLAG
ENSEMBLE_STAT_PROB_CAT_THRESH
ENSEMBLE_STAT_PROB_PCT_THRESH
ENSEMBLE_STAT_ECLV_POINTS
FCST_ENSEMBLE_STAT_IS_PROB
FCST_ENSEMBLE_STAT_PROB_IN_GRIB_PDS
ENSEMBLE_STAT_VERIFICATION_MASK_TEMPLATE
ENS_VAR<n>_NAME
ENS_VAR<n>_LEVELS
ENS_VAR<n>_THRESH
ENS_VAR<n>_OPTIONS
FCST_ENSEMBLE_STAT_VAR<n>_NAME
FCST_ENSEMBLE_STAT_VAR<n>_LEVELS
FCST_ENSEMBLE_STAT_VAR<n>_THRESH
FCST_ENSEMBLE_STAT_VAR<n>_OPTIONS
OBS_ENSEMBLE_STAT_VAR<n>_NAME
OBS_ENSEMBLE_STAT_VAR<n>_LEVELS
OBS_ENSEMBLE_STAT_VAR<n>_THRESH
OBS_ENSEMBLE_STAT_VAR<n>_OPTIONS

Warning: DEPRECATED:

ENSEMBLE_STAT_OUT_DIR
ENSEMBLE_STAT_CONFIG
ENSEMBLE_STAT_MET_OBS_ERROR_TABLE
ENSEMBLE_STAT_GRID_VX
ENSEMBLE_STAT_CLIMO_MEAN_INPUT_DIR
ENSEMBLE_STAT_CLIMO_STDEV_INPUT_DIR
ENSEMBLE_STAT_CLIMO_MEAN_INPUT_TEMPLATE
ENSEMBLE_STAT_CLIMO_STDEV_INPUT_TEMPLATE

6.3.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/EnsembleStatConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// Ensemble-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
${METPLUS_DESC}

//
// Output observation type to be written
//
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
```

(continues on next page)

(continued from previous page)

```
//
${METPLUS_CENSOR_THRESH}
${METPLUS_CENSOR_VAL}
cat_thresh    = [];
nc_var_str    = "";

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}


////////////////////////////////////////////////////////////////////////////////

//prob_cat_thresh =
${METPLUS_PROB_CAT_THRESH}

//prob_pct_thresh =
${METPLUS_PROB_PCT_THRESH}

//eclv_points =
${METPLUS_ECLV_POINTS}


////////////////////////////////////////////////////////////////////////////////

//
// Forecast and observation fields to be verified
//

fcst = {

   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_ENS_THRESH}
   ${METPLUS_VLD_THRESH}
   ${METPLUS_FCST_FIELD}
}

obs = {

   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}
}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//

${METPLUS_MESSAGE_TYPE}
sid_exc        = [];
//obs_thresh     = [ NA ];
${METPLUS_OBS_THRESH}

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

${METPLUS_DUPLICATE_FLAG}
obs_summary    = NONE;
obs_perc_value = 50;
${METPLUS_SKIP_CONST}

//
// Observation error options
// Set dist_type to NONE to use the observation error table instead
// May be set separately in each "obs.field" entry
//
obs_error = {
   ${METPLUS_OBS_ERROR_FLAG}
   dist_type        = NONE;
   dist_parm        = [];
   inst_bias_scale  = 1.0;
   inst_bias_offset = 0.0;
   min              = NA;      // Valid range of data
   max              = NA;
}

//
// Mapping of message type group name to comma-separated list of values.
//
message_type_group_map = [
   { key = "SURFACE"; val = "ADPSFC,SFCSHP,MSONET";               },
   { key = "ANYAIR";  val = "AIRCAR,AIRCFT";                      },
```

(continues on next page)

(continued from previous page)

```
   { key = "ANYSFC";  val = "ADPSFC,SFCSHP,ADPUPA,PROFLR,MSONET"; },
   { key = "ONLYSF";  val = "ADPSFC,SFCSHP";                      }
];

//
// Ensemble bin sizes
// May be set separately in each "obs.field" entry
//
${METPLUS_ENS_SSVAR_BIN_SIZE}
${METPLUS_ENS_PHIST_BIN_SIZE}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}




//
// May be set separately in each "obs.field" entry
//
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
mask = {
   ${METPLUS_MASK_GRID}
   ${METPLUS_MASK_POLY}
```

(continues on next page)

(continued from previous page)

```
   sid   = [];
   llpnt = [];
}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
${METPLUS_CI_ALPHA}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gridded verification output types
// May be set separately in each "obs.field" entry
//
${METPLUS_NC_ORANK_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Random number generator
//
rng = {
   type = "mt19937";
   seed = "1";
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

${METPLUS_OUTPUT_PREFIX}
//version          = "V9.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_DESC}

METplus Config(s)	MET Config File
DESC or *ENSEMBLE_STAT_DESC*	desc

${METPLUS_OBTYPE}

METplus Config(s)	MET Config File
OBTYPE	obtype

${METPLUS_REGRID_DICT}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_REGRID_SHAPE	regrid.shape
ENSEMBLE_STAT_REGRID_METHOD	regrid.method
ENSEMBLE_STAT_REGRID_WIDTH	regrid.width
ENSEMBLE_STAT_REGRID_VLD_THRESH	regrid.vld_thresh
ENSEMBLE_STAT_REGRID_TO_GRID	regrid.to_grid
ENSEMBLE_STAT_REGRID_CONVERT	regrid.convert
ENSEMBLE_STAT_REGRID_CENSOR_THRESH	regrid.censor_thresh
ENSEMBLE_STAT_REGRID_CENSOR_VAL	regrid.censor_val

${METPLUS_CENSOR_THRESH}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_CENSOR_THRESH	censor_thresh

${METPLUS_CENSOR_VAL}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_CENSOR_VAL	censor_val

${METPLUS_ENS_FILE_TYPE}

METplus Config(s)	MET Config File
ENS_ENSEMBLE_STAT_INPUT_DATATYPE	ens.file_type

${METPLUS_ENS_THRESH}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_ENS_THRESH	fcst.ens_thresh

${METPLUS_VLD_THRESH}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_VLD_THRESH	fcst.vld_thresh

${METPLUS_OBS_THRESH}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_OBS_THRESH	obs_thresh

${METPLUS_ENS_FIELD}

METplus Config(s)	MET Config File
ENS_VAR<n>_NAME	ens.field.name
ENS_VAR<n>_LEVELS	ens.field.level
ENS_VAR<n>_THRESH	ens.field.cat_thresh
ENS_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the forecast field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_PROB_CAT_THRESH}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_PROB_CAT_THRESH	prob_cat_thresh

${METPLUS_PROB_PCT_THRESH}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_PROB_PCT_THRESH	prob_pct_thresh

${METPLUS_ECLV_POINTS}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_ECLV_POINTS	eclv_points

${METPLUS_FCST_FILE_TYPE}

METplus Config(s)	MET Config File
FCST_ENSEMBLE_STAT_INPUT_DATATYPE	fcst.file_type

${METPLUS_FCST_FIELD}

METplus Config(s)	MET Config File
FCST_VAR<n>_NAME	fcst.field.name
FCST_VAR<n>_LEVELS	fcst.field.level
FCST_VAR<n>_THRESH	fcst.field.cat_thresh
FCST_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the forecast field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_OBS_FILE_TYPE}

METplus Config(s)	MET Config File
OBS_ENSEMBLE_STAT_INPUT_GRID_DATATYPE - or- *OBS_ENSEMBLE_STAT_INPUT_POINT_DATATYPE*	obs.file_type

${METPLUS_OBS_FIELD}

METplus Config(s)	MET Config File
OBS_VAR<n>_NAME	fcst.field.name
OBS_VAR<n>_LEVELS	fcst.field.level
OBS_VAR<n>_THRESH	fcst.field.cat_thresh
OBS_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the observation field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_MESSAGE_TYPE}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_MESSAGE_TYPE	message_type

${METPLUS_DUPLICATE_FLAG}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_DUPLICATE_FLAG	duplicate_flag

${METPLUS_SKIP_CONST}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_SKIP_CONST	skip_const

${METPLUS_OBS_ERROR_FLAG}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_OBS_ERROR_FLAG	obs_error.flag

${METPLUS_ENS_SSVAR_BIN_SIZE}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_ENS_SSVAR_BIN_SIZE	ens_ssvar_bin_size

${METPLUS_ENS_PHIST_BIN_SIZE}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_ENS_PHIST_BIN_SIZE	ens_phist_bin_size

${METPLUS_CLIMO_MEAN_DICT}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_CLIMO_MEAN_FILE_NAME	climo_mean.file_name
ENSEMBLE_STAT_CLIMO_MEAN_FIELD	climo_mean.field
ENSEMBLE_STAT_CLIMO_MEAN_REGRID_METHOD	climo_mean.regrid.method
ENSEMBLE_STAT_CLIMO_MEAN_REGRID_WIDTH	climo_mean.regrid.width
ENSEMBLE_STAT_CLIMO_MEAN_REGRID_VLD_THRESH	climo_mean.regrid.vld_thresh
ENSEMBLE_STAT_CLIMO_MEAN_REGRID_SHAPE	climo_mean.regrid.shape
ENSEMBLE_STAT_CLIMO_MEAN_TIME_INTERP_METHOD	climo_mean.time_interp_method
ENSEMBLE_STAT_CLIMO_MEAN_MATCH_MONTH	climo_mean.match_month
ENSEMBLE_STAT_CLIMO_MEAN_DAY_INTERVAL	climo_mean.day_interval
ENSEMBLE_STAT_CLIMO_MEAN_HOUR_INTERVAL	climo_mean.hour_interval

${METPLUS_CLIMO_STDEV_DICT}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_CLIMO_STDEV_FILE_NAME	climo_stdev.file_name
ENSEMBLE_STAT_CLIMO_STDEV_FIELD	climo_stdev.field
ENSEMBLE_STAT_CLIMO_STDEV_REGRID_METHOD	climo_stdev.regrid.method
ENSEMBLE_STAT_CLIMO_STDEV_REGRID_WIDTH	climo_stdev.regrid.width
ENSEMBLE_STAT_CLIMO_STDEV_REGRID_VLD_THRESH	climo_stdev.regrid.vld_thresh
ENSEMBLE_STAT_CLIMO_STDEV_REGRID_SHAPE	climo_stdev.regrid.shape
ENSEMBLE_STAT_CLIMO_STDEV_TIME_INTERP_METHOD	climo_stdev.time_interp_method
ENSEMBLE_STAT_CLIMO_STDEV_MATCH_MONTH	climo_stdev.match_month
ENSEMBLE_STAT_CLIMO_STDEV_DAY_INTERVAL	climo_stdev.day_interval
ENSEMBLE_STAT_CLIMO_STDEV_HOUR_INTERVAL	climo_stdev.hour_interval

${METPLUS_CLIMO_CDF_DICT}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_OBS_WINDOW_DICT}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_CLIMO_CDF_BINS	climo_cdv.cdf_bins
ENSEMBLE_STAT_CLIMO_CDF_CENTER_BINS	climo_cdv.center_bins
ENSEMBLE_STAT_CLIMO_CDF_WRITE_BINS	climo_cdv.write_bins
ENSEMBLE_STAT_CLIMO_CDF_DIRECT_PROB	climo_cdf.direct_prob

${METPLUS_MASK_GRID}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_MASK_GRID	mask.grid

${METPLUS_MASK_POLY}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_MASK_POLY	mask.poly

${METPLUS_CI_ALPHA}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_CI_ALPHA	ci_alpha

${METPLUS_INTERP_DICT}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_INTERP_FIELD	interp.field
ENSEMBLE_STAT_INTERP_VLD_THRESH	interp.vld_thresh
ENSEMBLE_STAT_INTERP_SHAPE	interp.shape
ENSEMBLE_STAT_INTERP_METHOD	interp.type.method
ENSEMBLE_STAT_INTERP_WIDTH	interp.type.width

${METPLUS_OUTPUT_FLAG_DICT}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_OUTPUT_FLAG_ECNT	output_flag.ecnt
ENSEMBLE_STAT_OUTPUT_FLAG_RPS	output_flag.rps
ENSEMBLE_STAT_OUTPUT_FLAG_RHIST	output_flag.rhist
ENSEMBLE_STAT_OUTPUT_FLAG_PHIST	output_flag.phist
ENSEMBLE_STAT_OUTPUT_FLAG_ORANK	output_flag.orank
ENSEMBLE_STAT_OUTPUT_FLAG_SSVAR	output_flag.ssvar
ENSEMBLE_STAT_OUTPUT_FLAG_RELP	output_flag.relp
ENSEMBLE_STAT_OUTPUT_FLAG_PCT	output_flag.pct
ENSEMBLE_STAT_OUTPUT_FLAG_PSTD	output_flag.pstd
ENSEMBLE_STAT_OUTPUT_FLAG_PJC	output_flag.pjc
ENSEMBLE_STAT_OUTPUT_FLAG_PRC	output_flag.prc
ENSEMBLE_STAT_OUTPUT_FLAG_ECLV	output_flag.eclv

${METPLUS_NC_ORANK_FLAG_DICT}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_NC_ORANK_FLAG_LATLON	nc_orank_flag.latlon
ENSEMBLE_STAT_NC_ORANK_FLAG_MEAN	nc_orank_flag.mean
ENSEMBLE_STAT_NC_ORANK_FLAG_RAW	nc_orank_flag.raw
ENSEMBLE_STAT_NC_ORANK_FLAG_RANK	nc_orank_flag.rank
ENSEMBLE_STAT_NC_ORANK_FLAG_PIT	nc_orank_flag.pit
ENSEMBLE_STAT_NC_ORANK_FLAG_VLD_COUNT	nc_orank_flag.vld_count
ENSEMBLE_STAT_NC_ORANK_FLAG_WEIGHT	nc_orank_flag.weight

${METPLUS_OUTPUT_PREFIX}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_OUTPUT_PREFIX	output_prefix

${METPLUS_OBS_QUALITY_INC}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_OBS_QUALITY_INC	obs_quality_inc

${METPLUS_OBS_QUALITY_EXC}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_OBS_QUALITY_EXC	obs_quality_exc

${METPLUS_ENS_MEMBER_IDS}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_ENS_MEMBER_IDS	ens_member_ids

${METPLUS_CONTROL_ID}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_CONTROL_ID	control_id

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_MET_CONFIG_OVERRIDES	n/a

${METPLUS_GRID_WEIGHT_FLAG}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_GRID_WEIGHT_FLAG	grid_weight_flag

6.4 Example

6.4.1 Description

Used to demonstrate how the METplus wrappers handle looping and building commands.

6.4.2 Configuration

EXAMPLE_INPUT_DIR
EXAMPLE_INPUT_TEMPLATE
EXAMPLE_CUSTOM_LOOP_LIST

6.5 ExtractTiles

6.5.1 Description

The ExtractTiles wrapper is used to regrid and extract subregions from paired tropical cyclone tracks generated with TCStat, or from cluster object centroids generated with MODE Time Domain (MTD). Unlike the other wrappers, the extract_tiles_wrapper does not correspond to a specific MET tool. It reads track information to determine the lat/lon positions of the paired track data. This information is then used to create tiles of subregions. The ExtractTiles wrapper creates a 2n degree x 2m degree grid/tile with each storm located at the center.

6.5.2 METplus Configuration

The following should be set in the METplus configuration file to define the dimensions and density of the tiles comprising the subregion:

EXTRACT_TILES_OUTPUT_DIR
EXTRACT_TILES_TC_STAT_INPUT_DIR
FCST_EXTRACT_TILES_INPUT_DIR
OBS_EXTRACT_TILES_INPUT_DIR
FCST_EXTRACT_TILES_INPUT_TEMPLATE
OBS_EXTRACT_TILES_INPUT_TEMPLATE
FCST_EXTRACT_TILES_OUTPUT_TEMPLATE
OBS_EXTRACT_TILES_OUTPUT_TEMPLATE
EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE
EXTRACT_TILES_MTD_INPUT_DIR
EXTRACT_TILES_MTD_INPUT_TEMPLATE
EXTRACT_TILES_LON_ADJ
EXTRACT_TILES_LAT_ADJ
EXTRACT_TILES_NLAT
EXTRACT_TILES_NLON
EXTRACT_TILES_DLON
EXTRACT_TILES_DLAT
EXTRACT_TILES_FILTER_OPTS
EXTRACT_TILES_VAR_LIST
EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS
EXTRACT_TILES_CUSTOM_LOOP_LIST

Warning: DEPRECATED:

EXTRACT_OUT_DIR
LON_ADJ
LAT_ADJ
NLAT
NLON
DLON
DLAT
EXTRACT_TILES_OVERWRITE_TRACK
EXTRACT_TILES_PAIRS_INPUT_DIR
EXTRACT_TILES_FILTERED_OUTPUT_TEMPLATE
EXTRACT_TILES_GRID_INPUT_DIR
EXTRACT_TILES_STAT_INPUT_DIR
EXTRACT_TILES_STAT_INPUT_TEMPLATE

6.6 GempakToCF

6.6.1 Description

Used to configure the utility GempakToCF.

6.6.2 METplus Configuration

GEMPAKTOCF_JAR
GEMPAKTOCF_INPUT_DIR
GEMPAKTOCF_OUTPUT_DIR
GEMPAKTOCF_INPUT_TEMPLATE
GEMPAKTOCF_OUTPUT_TEMPLATE
GEMPAKTOCF_SKIP_IF_OUTPUT_EXISTS
GEMPAKTOCF_CUSTOM_LOOP_LIST

Warning: DEPRECATED:

GEMPAKTOCF_CLASSPATH

6.7 GenEnsProd

6.7.1 Description

Used to configure the MET tool gen_ens_prod to generate ensemble products.

6.7.2 METplus Configuration

GEN_ENS_PROD_INPUT_DIR
GEN_ENS_PROD_INPUT_TEMPLATE
GEN_ENS_PROD_INPUT_FILE_LIST
GEN_ENS_PROD_CTRL_INPUT_DIR
GEN_ENS_PROD_CTRL_INPUT_TEMPLATE
GEN_ENS_PROD_OUTPUT_DIR
GEN_ENS_PROD_OUTPUT_TEMPLATE
LOG_GEN_ENS_PROD_VERBOSITY
MODEL
GEN_ENS_PROD_DESC
GEN_ENS_PROD_REGRID_TO_GRID
GEN_ENS_PROD_REGRID_METHOD
GEN_ENS_PROD_REGRID_WIDTH
GEN_ENS_PROD_REGRID_VLD_THRESH
GEN_ENS_PROD_REGRID_SHAPE
GEN_ENS_PROD_REGRID_CONVERT
GEN_ENS_PROD_REGRID_CENSOR_THRESH
GEN_ENS_PROD_REGRID_CENSOR_VAL
GEN_ENS_PROD_CENSOR_THRESH
GEN_ENS_PROD_CENSOR_VAL
GEN_ENS_PROD_CAT_THRESH
GEN_ENS_PROD_NORMALIZE
GEN_ENS_PROD_NC_VAR_STR
GEN_ENS_PROD_ENS_THRESH
GEN_ENS_PROD_ENS_VLD_THRESH
GEN_ENS_PROD_NBRHD_PROB_WIDTH
GEN_ENS_PROD_NBRHD_PROB_SHAPE
GEN_ENS_PROD_NBRHD_PROB_VLD_THRESH
GEN_ENS_PROD_NMEP_SMOOTH_VLD_THRESH

GEN_ENS_PROD_NMEP_SMOOTH_SHAPE
GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_DX
GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_RADIUS
GEN_ENS_PROD_NMEP_SMOOTH_METHOD
GEN_ENS_PROD_NMEP_SMOOTH_WIDTH
GEN_ENS_PROD_CLIMO_MEAN_FILE_NAME
GEN_ENS_PROD_CLIMO_MEAN_VAR<n>_NAME
GEN_ENS_PROD_CLIMO_MEAN_VAR<n>_LEVELS
GEN_ENS_PROD_CLIMO_MEAN_VAR<n>_OPTIONS
GEN_ENS_PROD_CLIMO_MEAN_FIELD
GEN_ENS_PROD_CLIMO_MEAN_REGRID_METHOD
GEN_ENS_PROD_CLIMO_MEAN_REGRID_WIDTH
GEN_ENS_PROD_CLIMO_MEAN_REGRID_VLD_THRESH
GEN_ENS_PROD_CLIMO_MEAN_REGRID_SHAPE
GEN_ENS_PROD_CLIMO_MEAN_TIME_INTERP_METHOD
GEN_ENS_PROD_CLIMO_MEAN_MATCH_MONTH
GEN_ENS_PROD_CLIMO_MEAN_DAY_INTERVAL
GEN_ENS_PROD_CLIMO_MEAN_HOUR_INTERVAL
GEN_ENS_PROD_CLIMO_MEAN_USE_FCST
GEN_ENS_PROD_CLIMO_MEAN_USE_OBS
GEN_ENS_PROD_CLIMO_STDEV_FILE_NAME
GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_NAME
GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_LEVELS
GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_OPTIONS
GEN_ENS_PROD_CLIMO_STDEV_FIELD
GEN_ENS_PROD_CLIMO_STDEV_REGRID_METHOD
GEN_ENS_PROD_CLIMO_STDEV_REGRID_WIDTH
GEN_ENS_PROD_CLIMO_STDEV_REGRID_VLD_THRESH
GEN_ENS_PROD_CLIMO_STDEV_REGRID_SHAPE
GEN_ENS_PROD_CLIMO_STDEV_TIME_INTERP_METHOD
GEN_ENS_PROD_CLIMO_STDEV_MATCH_MONTH
GEN_ENS_PROD_CLIMO_STDEV_DAY_INTERVAL
GEN_ENS_PROD_CLIMO_STDEV_HOUR_INTERVAL
GEN_ENS_PROD_CLIMO_STDEV_USE_FCST
GEN_ENS_PROD_CLIMO_STDEV_USE_OBS
GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON
GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN
GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV
GEN_ENS_PROD_ENSEMBLE_FLAG_MINUS
GEN_ENS_PROD_ENSEMBLE_FLAG_PLUS
GEN_ENS_PROD_ENSEMBLE_FLAG_MIN

GEN_ENS_PROD_ENSEMBLE_FLAG_MAX
GEN_ENS_PROD_ENSEMBLE_FLAG_RANGE
GEN_ENS_PROD_ENSEMBLE_FLAG_VLD_COUNT
GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY
GEN_ENS_PROD_ENSEMBLE_FLAG_NEP
GEN_ENS_PROD_ENSEMBLE_FLAG_NMEP
GEN_ENS_PROD_ENSEMBLE_FLAG_CLIMO
GEN_ENS_PROD_ENSEMBLE_FLAG_CLIMO_CDP
GEN_ENS_PROD_ENS_MEMBER_IDS
GEN_ENS_PROD_CONTROL_ID
GEN_ENS_PROD_MET_CONFIG_OVERRIDES

6.7.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/GenEnsProdConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// Gen-Ens-Prod configuration file.
//
// For additional information, please see the MET Users Guide.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
//desc =
${METPLUS_DESC}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
//censor_thresh =
${METPLUS_CENSOR_THRESH}

//censor_val    =
${METPLUS_CENSOR_VAL}

//normalize =
${METPLUS_NORMALIZE}

//cat_thresh    =
${METPLUS_CAT_THRESH}

//nc_var_str    =
${METPLUS_NC_VAR_STR}

//
// Ensemble fields to be processed
//
ens = {
   //file_type =
   ${METPLUS_ENS_FILE_TYPE}

   //ens_thresh =
   ${METPLUS_ENS_THRESH}

   //vld_thresh =
   ${METPLUS_VLD_THRESH}

   //field =
   ${METPLUS_ENS_FIELD}
```

(continues on next page)

(continued from previous page)

```
}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}


////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood ensemble probabilities
//
//nbrhd_prob = {
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
//nmep_smooth = {
${METPLUS_NMEP_SMOOTH_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}

//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Ensemble product output types
// May be set separately in each "ens.field" entry
//
//ensemble_flag = {
${METPLUS_ENSEMBLE_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//version = "V10.1.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_DESC}

METplus Config(s)	MET Config File
DESC or *GEN_ENS_PROD_DESC*	desc

${METPLUS_REGRID_DICT}

METplus Config(s)	MET Config File
GEN_ENS_PROD_REGRID_SHAPE	regrid.shape
GEN_ENS_PROD_REGRID_METHOD	regrid.method
GEN_ENS_PROD_REGRID_WIDTH	regrid.width
GEN_ENS_PROD_REGRID_VLD_THRESH	regrid.vld_thresh
GEN_ENS_PROD_REGRID_TO_GRID	regrid.to_grid
GEN_ENS_PROD_REGRID_CONVERT	regrid.convert
GEN_ENS_PROD_REGRID_CENSOR_THRESH	regrid.censor_thresh
GEN_ENS_PROD_REGRID_CENSOR_VAL	regrid.censor_val

${METPLUS_CENSOR_THRESH}

METplus Config(s)	MET Config File
GEN_ENS_PROD_CENSOR_THRESH	censor_thresh

${METPLUS_CENSOR_VAL}

METplus Config(s)	MET Config File
GEN_ENS_PROD_CENSOR_VAL	censor_val

${METPLUS_NORMALIZE}

METplus Config(s)	MET Config File
GEN_ENS_PROD_NORMALIZE	normalize

${METPLUS_CAT_THRESH}

METplus Config(s)	MET Config File
GEN_ENS_PROD_CAT_THRESH	cat_thresh

${METPLUS_NC_VAR_STR}

METplus Config(s)	MET Config File
GEN_ENS_PROD_NC_VAR_STR	nc_var_str

${METPLUS_ENS_FILE_TYPE}

METplus Config(s)	MET Config File
GEN_ENS_PROD_ENS_FILE_TYPE	ens.file_type

${METPLUS_ENS_ENS_THRESH}

METplus Config(s)	MET Config File
GEN_ENS_PROD_ENS_THRESH	ens.ens_thresh

${METPLUS_ENS_VLD_THRESH}

METplus Config(s)	MET Config File
GEN_ENS_PROD_ENS_VLD_THRESH	ens.vld_thresh

${METPLUS_ENS_FIELD}

METplus Config(s)	MET Config File
ENS_VAR<n>_NAME	ens.field.name
ENS_VAR<n>_LEVELS	ens.field.level
ENS_VAR<n>_THRESH	ens.field.cat_thresh
ENS_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the forecast field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_NBRHD_PROB_DICT}

METplus Config(s)	MET Config File
GEN_ENS_PROD_NBRHD_PROB_WIDTH	nbrhd_prob.width
GEN_ENS_PROD_NBRHD_PROB_SHAPE	nbrhd_prob.shape
GEN_ENS_PROD_NBRHD_PROB_VLD_THRESH	nbrhd_prob.vld_thresh

${METPLUS_NMEP_SMOOTH_DICT}

METplus Config(s)	MET Config File
GEN_ENS_PROD_NMEP_SMOOTH_VLD_THRESH	nmep_smooth.vld_thresh
GEN_ENS_PROD_NMEP_SMOOTH_SHAPE	nmep_smooth.shape
GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_DX	nmep_smooth.gaussian_dx
GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_RADIUS	nmep_smooth.gaussian_radius
GEN_ENS_PROD_NMEP_SMOOTH_METHOD	nmep_smooth.type.method
GEN_ENS_PROD_NMEP_SMOOTH_WIDTH	nmep_smooth.type.width

${METPLUS_CLIMO_MEAN_DICT}

METplus Config(s)	MET Config File
GEN_ENS_PROD_CLIMO_MEAN_FILE_NAME	climo_mean.file_name
GEN_ENS_PROD_CLIMO_MEAN_FIELD	climo_mean.field
GEN_ENS_PROD_CLIMO_MEAN_REGRID_METHOD	climo_mean.regrid.method
GEN_ENS_PROD_CLIMO_MEAN_REGRID_WIDTH	climo_mean.regrid.width
GEN_ENS_PROD_CLIMO_MEAN_REGRID_VLD_THRESH	climo_mean.regrid.vld_thresh
GEN_ENS_PROD_CLIMO_MEAN_REGRID_SHAPE	climo_mean.regrid.shape
GEN_ENS_PROD_CLIMO_MEAN_TIME_INTERP_METHOD	climo_mean.time_interp_method
GEN_ENS_PROD_CLIMO_MEAN_MATCH_MONTH	climo_mean.match_month
GEN_ENS_PROD_CLIMO_MEAN_DAY_INTERVAL	climo_mean.day_interval
GEN_ENS_PROD_CLIMO_MEAN_HOUR_INTERVAL	climo_mean.hour_interval

${METPLUS_CLIMO_STDEV_DICT}

METplus Config(s)	MET Config File
GEN_ENS_PROD_CLIMO_STDEV_FILE_NAME	climo_stdev.file_name
GEN_ENS_PROD_CLIMO_STDEV_FIELD	climo_stdev.field
GEN_ENS_PROD_CLIMO_STDEV_REGRID_METHOD	climo_stdev.regrid.method
GEN_ENS_PROD_CLIMO_STDEV_REGRID_WIDTH	climo_stdev.regrid.width
GEN_ENS_PROD_CLIMO_STDEV_REGRID_VLD_THRESH	climo_stdev.regrid.vld_thresh
GEN_ENS_PROD_CLIMO_STDEV_REGRID_SHAPE	climo_stdev.regrid.shape
GEN_ENS_PROD_CLIMO_STDEV_TIME_INTERP_METHOD	climo_stdev.time_interp_method
GEN_ENS_PROD_CLIMO_STDEV_MATCH_MONTH	climo_stdev.match_month
GEN_ENS_PROD_CLIMO_STDEV_DAY_INTERVAL	climo_stdev.day_interval
GEN_ENS_PROD_CLIMO_STDEV_HOUR_INTERVAL	climo_stdev.hour_interval

${METPLUS_ENSEMBLE_FLAG_DICT}

METplus Config(s)	MET Config File
GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON	ensemble_flag.latlon
GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN	ensemble_flag.mean
GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV	ensemble_flag.stdev
GEN_ENS_PROD_ENSEMBLE_FLAG_MINUS	ensemble_flag.minus
GEN_ENS_PROD_ENSEMBLE_FLAG_PLUS	ensemble_flag.plus
GEN_ENS_PROD_ENSEMBLE_FLAG_MIN	ensemble_flag.min
GEN_ENS_PROD_ENSEMBLE_FLAG_MAX	ensemble_flag.max
GEN_ENS_PROD_ENSEMBLE_FLAG_RANGE	ensemble_flag.range
GEN_ENS_PROD_ENSEMBLE_FLAG_VLD_COUNT	ensemble_flag.vld_count
GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY	ensemble_flag.frequency
GEN_ENS_PROD_ENSEMBLE_FLAG_NEP	ensemble_flag.nep
GEN_ENS_PROD_ENSEMBLE_FLAG_NMEP	ensemble_flag.nmep
GEN_ENS_PROD_ENSEMBLE_FLAG_CLIMO	ensemble_flag.climo
GEN_ENS_PROD_ENSEMBLE_FLAG_CLIMO_CDP	ensemble_flag.climo_cdp

${METPLUS_ENS_MEMBER_IDS}

METplus Config(s)	MET Config File
GEN_ENS_PROD_ENS_MEMBER_IDS	ens_member_ids

${METPLUS_CONTROL_ID}

METplus Config(s)	MET Config File
GEN_ENS_PROD_CONTROL_ID	control_id

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
GEN_ENS_PROD_MET_CONFIG_OVERRIDES	n/a

6.8 GenVxMask

6.8.1 Description

Used to configure the MET tool GenVxMask to define and generate masking regions.

6.8.2 Configuration

GEN_VX_MASK_INPUT_DIR
GEN_VX_MASK_INPUT_MASK_DIR
GEN_VX_MASK_OUTPUT_DIR
GEN_VX_MASK_INPUT_TEMPLATE
GEN_VX_MASK_INPUT_MASK_TEMPLATE
GEN_VX_MASK_OUTPUT_TEMPLATE
GEN_VX_MASK_OPTIONS
LOG_GEN_VX_MASK_VERBOSITY
GEN_VX_MASK_SKIP_IF_OUTPUT_EXISTS
GEN_VX_MASK_CUSTOM_LOOP_LIST
GEN_VX_MASK_FILE_WINDOW_BEGIN
GEN_VX_MASK_FILE_WINDOW_END

6.9 GFDLTracker

6.9.1 Description

Used to call the GFDL Tracker applications to objectively analyze forecast data to provide an estimate of the vortex center position (latitude and longitude), and track the storm for the duration of the forecast. The wrapper copies files and uses symbolic links to ensure that input files are named and located in the correct place so that the tracker can read them. The wrapper also generates index files and other inputs that are required to run the tool and substitutes values into template configuration files that are read by the tracker. Relevant output files are renamed based on user configuration. See *GFDL Tracker (optional)* (page 34) for more information.

6.9.2 METplus Configuration

GFDL_TRACKER_BASE
GFDL_TRACKER_INPUT_DIR
GFDL_TRACKER_INPUT_TEMPLATE
GFDL_TRACKER_TC_VITALS_INPUT_DIR
GFDL_TRACKER_TC_VITALS_INPUT_TEMPLATE
GFDL_TRACKER_OUTPUT_DIR
GFDL_TRACKER_OUTPUT_TEMPLATE
GFDL_TRACKER_GRIB_VERSION
GFDL_TRACKER_NML_TEMPLATE_FILE
GFDL_TRACKER_DATEIN_INP_MODEL
GFDL_TRACKER_DATEIN_INP_MODTYP
GFDL_TRACKER_DATEIN_INP_LT_UNITS

GFDL_TRACKER_DATEIN_INP_FILE_SEQ
GFDL_TRACKER_DATEIN_INP_NESTTYP
GFDL_TRACKER_ATCFINFO_ATCFNUM
GFDL_TRACKER_ATCFINFO_ATCFNAME
GFDL_TRACKER_ATCFINFO_ATCFFREQ
GFDL_TRACKER_TRACKERINFO_TYPE
GFDL_TRACKER_TRACKERINFO_MSLPTHRESH
GFDL_TRACKER_TRACKERINFO_USE_BACKUP_MSLP_GRAD_CHECK
GFDL_TRACKER_TRACKERINFO_V850THRESH
GFDL_TRACKER_TRACKERINFO_USE_BACKUP_850_VT_CHECK
GFDL_TRACKER_TRACKERINFO_ENABLE_TIMING
GFDL_TRACKER_TRACKERINFO_GRIDTYPE
GFDL_TRACKER_TRACKERINFO_CONTINT
GFDL_TRACKER_TRACKERINFO_WANT_OCI
GFDL_TRACKER_TRACKERINFO_OUT_VIT
GFDL_TRACKER_TRACKERINFO_USE_LAND_MASK
GFDL_TRACKER_TRACKERINFO_INP_DATA_TYPE
GFDL_TRACKER_TRACKERINFO_GRIBVER
GFDL_TRACKER_TRACKERINFO_G2_JPDTN
GFDL_TRACKER_TRACKERINFO_G2_MSLP_PARM_ID
GFDL_TRACKER_TRACKERINFO_G1_MSLP_PARM_ID
GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_TYP
GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_VAL
GFDL_TRACKER_PHASEINFO_PHASEFLAG
GFDL_TRACKER_PHASEINFO_PHASESCHEME
GFDL_TRACKER_PHASEINFO_WCORE_DEPTH
GFDL_TRACKER_STRUCTINFO_STRUCTFLAG
GFDL_TRACKER_STRUCTINFO_IKEFLAG
GFDL_TRACKER_FNAMEINFO_GMODNAME
GFDL_TRACKER_FNAMEINFO_RUNDESCR
GFDL_TRACKER_FNAMEINFO_ATCFDESCR
GFDL_TRACKER_WAITINFO_USE_WAITFOR
GFDL_TRACKER_WAITINFO_WAIT_MIN_AGE
GFDL_TRACKER_WAITINFO_WAIT_MIN_SIZE
GFDL_TRACKER_WAITINFO_WAIT_MAX_WAIT
GFDL_TRACKER_WAITINFO_WAIT_SLEEPTIME
GFDL_TRACKER_WAITINFO_USE_PER_FCST_COMMAND
GFDL_TRACKER_WAITINFO_PER_FCST_COMMAND
GFDL_TRACKER_NETCDFINFO_LAT_NAME
GFDL_TRACKER_NETCDFINFO_LMASKNAME
GFDL_TRACKER_NETCDFINFO_LON_NAME

GFDL_TRACKER_NETCDFINFO_MSLPNAME
GFDL_TRACKER_NETCDFINFO_NETCDF_FILENAME
GFDL_TRACKER_NETCDFINFO_NUM_NETCDF_VARS
GFDL_TRACKER_NETCDFINFO_RV700NAME
GFDL_TRACKER_NETCDFINFO_RV850NAME
GFDL_TRACKER_NETCDFINFO_TIME_NAME
GFDL_TRACKER_NETCDFINFO_TIME_UNITS
GFDL_TRACKER_NETCDFINFO_TMEAN_300_500_NAME
GFDL_TRACKER_NETCDFINFO_U500NAME
GFDL_TRACKER_NETCDFINFO_U700NAME
GFDL_TRACKER_NETCDFINFO_U850NAME
GFDL_TRACKER_NETCDFINFO_USFCNAME
GFDL_TRACKER_NETCDFINFO_V500NAME
GFDL_TRACKER_NETCDFINFO_V700NAME
GFDL_TRACKER_NETCDFINFO_V850NAME
GFDL_TRACKER_NETCDFINFO_VSFCNAME
GFDL_TRACKER_NETCDFINFO_Z200NAME
GFDL_TRACKER_NETCDFINFO_Z300NAME
GFDL_TRACKER_NETCDFINFO_Z350NAME
GFDL_TRACKER_NETCDFINFO_Z400NAME
GFDL_TRACKER_NETCDFINFO_Z450NAME
GFDL_TRACKER_NETCDFINFO_Z500NAME
GFDL_TRACKER_NETCDFINFO_Z550NAME
GFDL_TRACKER_NETCDFINFO_Z600NAME
GFDL_TRACKER_NETCDFINFO_Z650NAME
GFDL_TRACKER_NETCDFINFO_Z700NAME
GFDL_TRACKER_NETCDFINFO_Z750NAME
GFDL_TRACKER_NETCDFINFO_Z800NAME
GFDL_TRACKER_NETCDFINFO_Z850NAME
GFDL_TRACKER_NETCDFINFO_Z900NAME
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA700
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC850
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC700
GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH850
GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH700
GFDL_TRACKER_USER_WANTS_TO_TRACK_MSLP
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRCSFC
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETASFC
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK500850
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200500
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200850

GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA850
GFDL_TRACKER_VERBOSE_VERB
GFDL_TRACKER_VERBOSE_VERB_G2
GFDL_TRACKER_KEEP_INTERMEDIATE

6.9.3 NML Configuration

Below is the NML template configuration file used for this wrapper. The wrapper substitutes values from the METplus configuration file into this configuration file. While it may appear that environment variables are used in the NML template file, they are not actually environment variables. The wrapper searches for these strings and substitutes the values as appropriate.

```
&datein
  inp%bcc = ${METPLUS_DATEIN_INP_BCC},
  inp%byy = ${METPLUS_DATEIN_INP_BYY},
  inp%bmm = ${METPLUS_DATEIN_INP_BMM},
  inp%bdd = ${METPLUS_DATEIN_INP_BDD},
  inp%bhh = ${METPLUS_DATEIN_INP_BHH},
  inp%model = ${METPLUS_DATEIN_INP_MODEL},
  inp%modtyp = ${METPLUS_DATEIN_INP_MODTYP},
  inp%lt_units = ${METPLUS_DATEIN_INP_LT_UNITS},
  inp%file_seq = ${METPLUS_DATEIN_INP_FILE_SEQ},
  inp%nesttyp = ${METPLUS_DATEIN_INP_NESTTYP},
/

&atcfinfo
  atcfnum = ${METPLUS_ATCFINFO_ATCFNUM},
  atcfname = ${METPLUS_ATCFINFO_ATCFNAME},
  atcfymdh = ${METPLUS_ATCFINFO_ATCFYMDH},
  atcffreq = ${METPLUS_ATCFINFO_ATCFFREQ},
/

&trackerinfo
  trkrinfo%type = ${METPLUS_TRACKERINFO_TYPE},
  trkrinfo%mslpthresh = ${METPLUS_TRACKERINFO_MSLPTHRESH},
  trkrinfo%use_backup_mslp_grad_check = ${METPLUS_TRACKERINFO_USE_BACKUP_MSLP_GRAD_CHECK},
  trkrinfo%v850thresh = ${METPLUS_TRACKERINFO_V850THRESH},
  trkrinfo%use_backup_850_vt_check = ${METPLUS_TRACKERINFO_USE_BACKUP_850_VT_CHECK},
  trkrinfo%enable_timing = ${METPLUS_TRACKERINFO_ENABLE_TIMING},
  trkrinfo%gridtype = ${METPLUS_TRACKERINFO_GRIDTYPE},
  trkrinfo%contint = ${METPLUS_TRACKERINFO_CONTINT},
  trkrinfo%want_oci = ${METPLUS_TRACKERINFO_WANT_OCI},
  trkrinfo%out_vit = ${METPLUS_TRACKERINFO_OUT_VIT},
  trkrinfo%use_land_mask = ${METPLUS_TRACKERINFO_USE_LAND_MASK},
  trkrinfo%inp_data_type = ${METPLUS_TRACKERINFO_INP_DATA_TYPE},
```

(continues on next page)

(continued from previous page)

```
  trkrinfo%gribver = ${METPLUS_TRACKERINFO_GRIBVER},
  trkrinfo%g2_jpdtn = ${METPLUS_TRACKERINFO_G2_JPDTN},
  trkrinfo%g2_mslp_parm_id = ${METPLUS_TRACKERINFO_G2_MSLP_PARM_ID},
  trkrinfo%g1_mslp_parm_id = ${METPLUS_TRACKERINFO_G1_MSLP_PARM_ID},
  trkrinfo%g1_sfcwind_lev_typ = ${METPLUS_TRACKERINFO_G1_SFCWIND_LEV_TYP},
  trkrinfo%g1_sfcwind_lev_val = ${METPLUS_TRACKERINFO_G1_SFCWIND_LEV_VAL},
  trkrinfo%westbd = ${METPLUS_TRACKERINFO_WESTBD},
  trkrinfo%eastbd = ${METPLUS_TRACKERINFO_EASTBD},
  trkrinfo%southbd = ${METPLUS_TRACKERINFO_SOUTHBD},
  trkrinfo%northbd = ${METPLUS_TRACKERINFO_NORTHBD},
/

&phaseinfo
  phaseflag = ${METPLUS_PHASEINFO_PHASEFLAG},
  phasescheme = ${METPLUS_PHASEINFO_PHASESCHEME},
  wcore_depth = ${METPLUS_PHASEINFO_WCORE_DEPTH},
/

&structinfo
  structflag = ${METPLUS_STRUCTINFO_STRUCTFLAG},
  ikeflag = ${METPLUS_STRUCTINFO_IKEFLAG},
/

&fnameinfo
  gmodname = ${METPLUS_FNAMEINFO_GMODNAME},
  rundescr = ${METPLUS_FNAMEINFO_RUNDESCR},
  atcfdescr = ${METPLUS_FNAMEINFO_ATCFDESCR},
/

&waitinfo
  use_waitfor = ${METPLUS_WAITINFO_USE_WAITFOR},
  wait_min_age = ${METPLUS_WAITINFO_WAIT_MIN_AGE},
  wait_min_size = ${METPLUS_WAITINFO_WAIT_MIN_SIZE},
  wait_max_wait = ${METPLUS_WAITINFO_WAIT_MAX_WAIT},
  wait_sleeptime = ${METPLUS_WAITINFO_WAIT_SLEEPTIME},
  use_per_fcst_command = ${METPLUS_WAITINFO_USE_PER_FCST_COMMAND},
  per_fcst_command = ${METPLUS_WAITINFO_PER_FCST_COMMAND},
/

&netcdflist
  netcdfinfo%lat_name = ${METPLUS_NETCDFINFO_LAT_NAME},
  netcdfinfo%lmaskname = ${METPLUS_NETCDFINFO_LMASKNAME},
  netcdfinfo%lon_name = ${METPLUS_NETCDFINFO_LON_NAME},
  netcdfinfo%mslpname = ${METPLUS_NETCDFINFO_MSLPNAME},
  netcdfinfo%netcdf_filename = ${METPLUS_NETCDFINFO_NETCDF_FILENAME},
```

(continues on next page)

(continued from previous page)

```
  netcdfinfo%num_netcdf_vars = ${METPLUS_NETCDFINFO_NUM_NETCDF_VARS},
  netcdfinfo%rv700name = ${METPLUS_NETCDFINFO_RV700NAME},
  netcdfinfo%rv850name = ${METPLUS_NETCDFINFO_RV850NAME},
  netcdfinfo%time_name = ${METPLUS_NETCDFINFO_TIME_NAME},
  netcdfinfo%time_units = ${METPLUS_NETCDFINFO_TIME_UNITS},
  netcdfinfo%tmean_300_500_name = ${METPLUS_NETCDFINFO_TMEAN_300_500_NAME},
  netcdfinfo%u500name = ${METPLUS_NETCDFINFO_U500NAME},
  netcdfinfo%u700name = ${METPLUS_NETCDFINFO_U700NAME},
  netcdfinfo%u850name = ${METPLUS_NETCDFINFO_U850NAME},
  netcdfinfo%usfcname = ${METPLUS_NETCDFINFO_USFCNAME},
  netcdfinfo%v500name = ${METPLUS_NETCDFINFO_V500NAME},
  netcdfinfo%v700name = ${METPLUS_NETCDFINFO_V700NAME},
  netcdfinfo%v850name = ${METPLUS_NETCDFINFO_V850NAME},
  netcdfinfo%vsfcname = ${METPLUS_NETCDFINFO_VSFCNAME},
  netcdfinfo%z200name = ${METPLUS_NETCDFINFO_Z200NAME},
  netcdfinfo%z300name = ${METPLUS_NETCDFINFO_Z300NAME},
  netcdfinfo%z350name = ${METPLUS_NETCDFINFO_Z350NAME},
  netcdfinfo%z400name = ${METPLUS_NETCDFINFO_Z400NAME},
  netcdfinfo%z450name = ${METPLUS_NETCDFINFO_Z450NAME},
  netcdfinfo%z500name = ${METPLUS_NETCDFINFO_Z500NAME},
  netcdfinfo%z550name = ${METPLUS_NETCDFINFO_Z550NAME},
  netcdfinfo%z600name = ${METPLUS_NETCDFINFO_Z600NAME},
  netcdfinfo%z650name = ${METPLUS_NETCDFINFO_Z650NAME},
  netcdfinfo%z700name = ${METPLUS_NETCDFINFO_Z700NAME},
  netcdfinfo%z750name = ${METPLUS_NETCDFINFO_Z750NAME},
  netcdfinfo%z800name = ${METPLUS_NETCDFINFO_Z800NAME},
  netcdfinfo%z850name = ${METPLUS_NETCDFINFO_Z850NAME},
  netcdfinfo%z900name = ${METPLUS_NETCDFINFO_Z900NAME},
/

&parmpreflist
  user_wants_to_track_zeta700 = ${METPLUS_USER_WANTS_TO_TRACK_ZETA700},
  user_wants_to_track_wcirc850 = ${METPLUS_USER_WANTS_TO_TRACK_WCIRC850},
  user_wants_to_track_wcirc700 = ${METPLUS_USER_WANTS_TO_TRACK_WCIRC700},
  user_wants_to_track_gph850 = ${METPLUS_USER_WANTS_TO_TRACK_GPH850},
  user_wants_to_track_gph700 = ${METPLUS_USER_WANTS_TO_TRACK_GPH700},
  user_wants_to_track_mslp = ${METPLUS_USER_WANTS_TO_TRACK_MSLP},
  user_wants_to_track_wcircsfc = ${METPLUS_USER_WANTS_TO_TRACK_WCIRCSFC},
  user_wants_to_track_zetasfc = ${METPLUS_USER_WANTS_TO_TRACK_ZETASFC},
  user_wants_to_track_thick500850 = ${METPLUS_USER_WANTS_TO_TRACK_THICK500850},
  user_wants_to_track_thick200500 = ${METPLUS_USER_WANTS_TO_TRACK_THICK200500},
  user_wants_to_track_thick200850 = ${METPLUS_USER_WANTS_TO_TRACK_THICK200850},
  user_wants_to_track_zeta850 = ${METPLUS_USER_WANTS_TO_TRACK_ZETA850},
/
```

(continues on next page)

(continued from previous page)

```
&verbose
  verb = ${METPLUS_VERBOSE_VERB},
  verb_g2 = ${METPLUS_VERBOSE_VERB_G2},
/
```

${METPLUS_DATEIN_INP_BCC}

METplus Config(s)	NML Config File
INIT_BEG	&datein: inp%bcc

${METPLUS_DATEIN_INP_BYY}

METplus Config(s)	NML Config File
INIT_BEG	&datein: inp%byy

${METPLUS_DATEIN_INP_BMM}

METplus Config(s)	NML Config File
INIT_BEG	&datein: inp%bmm

${METPLUS_DATEIN_INP_BDD}

METplus Config(s)	NML Config File
INIT_BEG	&datein: inp%bdd

${METPLUS_DATEIN_INP_BHH}

METplus Config(s)	NML Config File
INIT_BEG	&datein: inp%bhh

${METPLUS_DATEIN_INP_MODEL}

METplus Config(s)	NML Config File
GFDL_TRACKER_DATEIN_INP_MODEL	&datein: inp%model

${METPLUS_DATEIN_INP_MODTYP}

METplus Config(s)	NML Config File
GFDL_TRACKER_DATEIN_INP_MODTYP	&datein: inp%modtyp

${METPLUS_DATEIN_INP_LT_UNITS}

METplus Config(s)	NML Config File
GFDL_TRACKER_DATEIN_INP_LT_UNITS	&datein: inp%lt_units

${METPLUS_DATEIN_INP_FILE_SEQ}

METplus Config(s)	NML Config File
GFDL_TRACKER_DATEIN_INP_FILE_SEQ	&datein: inp%file_seq

${METPLUS_DATEIN_INP_NESTTYP}

METplus Config(s)	NML Config File
GFDL_TRACKER_DATEIN_INP_NESTTYP	&datein: inp%nesttyp

${METPLUS_ATCFINFO_ATCFNUM}

METplus Config(s)	NML Config File
GFDL_TRACKER_ATCFINFO_ATCFNUM	&atcfinfo: atcfnum

${METPLUS_ATCFINFO_ATCFNAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_ATCFINFO_ATCFNAME	&atcfinfo: atcfname

${METPLUS_ATCFINFO_ATCFYMDH}

METplus Config(s)	NML Config File
INIT_BEG	&atcfinfo: atcfymdh

${METPLUS_ATCFINFO_ATCFFREQ}

METplus Config(s)	NML Config File
GFDL_TRACKER_ATCFINFO_ATCFFREQ	&atcfinfo: atcffreq

${METPLUS_TRACKERINFO_TYPE}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_TYPE	&trackerinfo: trkrinfo%type

${METPLUS_TRACKERINFO_MSLPTHRESH}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_MSLPTHRESH	&trackerinfo: trkrinfo%mslpthresh

${METPLUS_TRACKERINFO_USE_BACKUP_MSLP_GRAD_CHECK}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_USE_BACKUP_MSLP_GRAD_CHECK	&trackerinfo: trkrinfo%use_backup_mslp_grad_check

${METPLUS_TRACKERINFO_V850THRESH}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_V850THRESH	&trackerinfo: trkrinfo%v850thresh

${METPLUS_TRACKERINFO_USE_BACKUP_850_VT_CHECK}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_USE_BACKUP_850_VT_CHECK	&trackerinfo: trkrinfo%use_backup_850_vt_check

${METPLUS_TRACKERINFO_ENABLE_TIMING}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_ENABLE_TIMING	&trackerinfo: trkrinfo%enable_timing

${METPLUS_TRACKERINFO_GRIDTYPE}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_GRIDTYPE	&trackerinfo: trkrinfo%gridtype

${METPLUS_TRACKERINFO_CONTINT}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_CONTINT	&trackerinfo: trkrinfo%contint

${METPLUS_TRACKERINFO_WANT_OCI}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_WANT_OCI	&trackerinfo: trkrinfo%want_oci

${METPLUS_TRACKERINFO_OUT_VIT}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_OUT_VIT	&trackerinfo: trkrinfo%out_vit

${METPLUS_TRACKERINFO_USE_LAND_MASK}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_USE_LAND_MASK	&trackerinfo: trkrinfo%use_land_mask

${METPLUS_TRACKERINFO_INP_DATA_TYPE}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_INP_DATA_TYPE	&trackerinfo: trkrinfo%inp_data_type

${METPLUS_TRACKERINFO_GRIBVER}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_GRIBVER	&trackerinfo: trkrinfo%gribver

${METPLUS_TRACKERINFO_G2_JPDTN}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_G2_JPDTN	&trackerinfo: trkrinfo%g2_jpdtn

${METPLUS_TRACKERINFO_G2_MSLP_PARM_ID}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_G2_MSLP_PARM_ID	&trackerinfo: trkrinfo%g2_mslp_parm_id

${METPLUS_TRACKERINFO_G1_MSLP_PARM_ID}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_G1_MSLP_PARM_ID	&trackerinfo: trkrinfo%g1_mslp_parm_id

${METPLUS_TRACKERINFO_G1_SFCWIND_LEV_TYP}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_TYP	&trackerinfo: trkrinfo%g1_sfcwind_lev_typ

${METPLUS_TRACKERINFO_G1_SFCWIND_LEV_VAL}

METplus Config(s)	NML Config File
GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_VAL	&trackerinfo: trkrinfo%g1_sfcwind_lev_val

${METPLUS_PHASEINFO_PHASEFLAG}

METplus Config(s)	NML Config File
GFDL_TRACKER_PHASEINFO_PHASEFLAG	&phaseinfo: phaseflag

${METPLUS_PHASEINFO_PHASESCHEME}

METplus Config(s)	NML Config File
GFDL_TRACKER_PHASEINFO_PHASESCHEME	&phaseinfo: phasescheme

${METPLUS_PHASEINFO_WCORE_DEPTH}

METplus Config(s)	NML Config File
GFDL_TRACKER_PHASEINFO_WCORE_DEPTH	&phaseinfo: wcore_depth

${METPLUS_STRUCTINFO_STRUCTFLAG}

METplus Config(s)	NML Config File
GFDL_TRACKER_STRUCTINFO_STRUCTFLAG	&structinfo: structflag

${METPLUS_STRUCTINFO_IKEFLAG}

METplus Config(s)	NML Config File
GFDL_TRACKER_STRUCTINFO_IKEFLAG	&structinfo: ikeflag

${METPLUS_FNAMEINFO_GMODNAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_FNAMEINFO_GMODNAME	&fnameinfo: gmodname

${METPLUS_FNAMEINFO_RUNDESCR}

METplus Config(s)	NML Config File
GFDL_TRACKER_FNAMEINFO_RUNDESCR	&fnameinfo: rundescr

${METPLUS_FNAMEINFO_ATCFDESCR}

METplus Config(s)	NML Config File
GFDL_TRACKER_FNAMEINFO_ATCFDESCR	&fnameinfo: atcfdescr

${METPLUS_WAITINFO_USE_WAITFOR}

METplus Config(s)	NML Config File
GFDL_TRACKER_WAITINFO_USE_WAITFOR	&waitinfo: use_waitfor

${METPLUS_WAITINFO_WAIT_MIN_AGE}

METplus Config(s)	NML Config File
GFDL_TRACKER_WAITINFO_WAIT_MIN_AGE	&waitinfo: wait_min_age

${METPLUS_WAITINFO_WAIT_MIN_SIZE}

METplus Config(s)	NML Config File
GFDL_TRACKER_WAITINFO_WAIT_MIN_SIZE	&waitinfo: wait_min_size

${METPLUS_WAITINFO_WAIT_MAX_WAIT}

METplus Config(s)	NML Config File
GFDL_TRACKER_WAITINFO_WAIT_MAX_WAIT	&waitinfo: wait_max_wait

${METPLUS_WAITINFO_WAIT_SLEEPTIME}

METplus Config(s)	NML Config File
GFDL_TRACKER_WAITINFO_WAIT_SLEEPTIME	&waitinfo: wait_sleeptime

${METPLUS_WAITINFO_USE_PER_FCST_COMMAND}

METplus Config(s)	NML Config File
GFDL_TRACKER_WAITINFO_USE_PER_FCST_COMMAND	&waitinfo: use_per_fcst_command

${METPLUS_WAITINFO_PER_FCST_COMMAND}

METplus Config(s)	NML Config File
GFDL_TRACKER_WAITINFO_PER_FCST_COMMAND	&waitinfo: per_fcst_command

${METPLUS_NETCDFINFO_LAT_NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_LAT_NAME	&netcdflist: netcdfinfo%lat_name

${METPLUS_NETCDFINFO_LMASKNAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_LMASKNAME	&netcdflist: netcdfinfo%lmaskname

${METPLUS_NETCDFINFO_LON_NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_LON_NAME	&netcdflist: netcdfinfo%lon_name

${METPLUS_NETCDFINFO_MSLPNAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_MSLPNAME	&netcdflist: netcdfinfo%mslpname

${METPLUS_NETCDFINFO_NETCDF_FILENAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_NETCDF_FILENAME	&netcdflist: netcdfinfo%netcdf_filename

${METPLUS_NETCDFINFO_NUM_NETCDF_VARS}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_NUM_NETCDF_VARS	&netcdflist: netcdfinfo%num_netcdf_vars

${METPLUS_NETCDFINFO_RV700NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_RV700NAME	&netcdflist: netcdfinfo%rv700name

${METPLUS_NETCDFINFO_RV850NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_RV850NAME	&netcdflist: netcdfinfo%rv850name

${METPLUS_NETCDFINFO_TIME_NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_TIME_NAME	&netcdflist: netcdfinfo%time_name

${METPLUS_NETCDFINFO_TIME_UNITS}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_TIME_UNITS	&netcdflist: netcdfinfo%time_units

${METPLUS_NETCDFINFO_TMEAN_300_500_NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_TMEAN_300_500_NAME	&netcdflist: netcdfinfo%tmean_300_500_name

${METPLUS_NETCDFINFO_U500NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_U500NAME	&netcdflist: netcdfinfo%u500name

${METPLUS_NETCDFINFO_U700NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_U700NAME	&netcdflist: netcdfinfo%u700name

${METPLUS_NETCDFINFO_U850NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_U850NAME	&netcdflist: netcdfinfo%u850name

${METPLUS_NETCDFINFO_USFCNAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_USFCNAME	&netcdflist: netcdfinfo%usfcname

${METPLUS_NETCDFINFO_V500NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_V500NAME	&netcdflist: netcdfinfo%v500name

${METPLUS_NETCDFINFO_V700NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_V700NAME	&netcdflist: netcdfinfo%v700name

${METPLUS_NETCDFINFO_V850NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_V850NAME	&netcdflist: netcdfinfo%v850name

${METPLUS_NETCDFINFO_VSFCNAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_VSFCNAME	&netcdflist: netcdfinfo%vsfcname

${METPLUS_NETCDFINFO_Z200NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z200NAME	&netcdflist: netcdfinfo%z200name

${METPLUS_NETCDFINFO_Z300NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z300NAME	&netcdflist: netcdfinfo%z300name

${METPLUS_NETCDFINFO_Z350NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z350NAME	&netcdflist: netcdfinfo%z350name

${METPLUS_NETCDFINFO_Z400NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z400NAME	&netcdflist: netcdfinfo%z400name

${METPLUS_NETCDFINFO_Z450NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z450NAME	&netcdflist: netcdfinfo%z450name

${METPLUS_NETCDFINFO_Z500NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z500NAME	&netcdflist: netcdfinfo%z500name

${METPLUS_NETCDFINFO_Z550NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z550NAME	&netcdflist: netcdfinfo%z550name

${METPLUS_NETCDFINFO_Z600NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z600NAME	&netcdflist: netcdfinfo%z600name

${METPLUS_NETCDFINFO_Z650NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z650NAME	&netcdflist: netcdfinfo%z650name

${METPLUS_NETCDFINFO_Z700NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z700NAME	&netcdflist: netcdfinfo%z700name

${METPLUS_NETCDFINFO_Z750NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z750NAME	&netcdflist: netcdfinfo%z750name

${METPLUS_NETCDFINFO_Z800NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z800NAME	&netcdflist: netcdfinfo%z800name

${METPLUS_NETCDFINFO_Z850NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z850NAME	&netcdflist: netcdfinfo%z850name

${METPLUS_NETCDFINFO_Z900NAME}

METplus Config(s)	NML Config File
GFDL_TRACKER_NETCDFINFO_Z900NAME	&netcdflist: netcdfinfo%z900name

${METPLUS_USER_WANTS_TO_TRACK_ZETA700}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA700	&parmpreflist: user_wants_to_track_zeta700

${METPLUS_USER_WANTS_TO_TRACK_WCIRC850}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC850	&parmpreflist: user_wants_to_track_wcirc850

${METPLUS_USER_WANTS_TO_TRACK_WCIRC700}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC700	&parmpreflist: user_wants_to_track_wcirc700

${METPLUS_USER_WANTS_TO_TRACK_GPH850}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH850	&parmpreflist: user_wants_to_track_gph850

${METPLUS_USER_WANTS_TO_TRACK_GPH700}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH700	&parmpreflist: user_wants_to_track_gph700

${METPLUS_USER_WANTS_TO_TRACK_MSLP}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_MSLP	&parmpreflist: user_wants_to_track_mslp

${METPLUS_USER_WANTS_TO_TRACK_WCIRCSFC}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRCSFC	&parmpreflist: user_wants_to_track_wcircsfc

${METPLUS_USER_WANTS_TO_TRACK_ZETASFC}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETASFC	&parmpreflist: user_wants_to_track_zetasfc

${METPLUS_USER_WANTS_TO_TRACK_THICK500850}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK500850	&parmpreflist: user_wants_to_track_thick500850

${METPLUS_USER_WANTS_TO_TRACK_THICK200500}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200500	&parmpreflist: user_wants_to_track_thick200500

${METPLUS_USER_WANTS_TO_TRACK_THICK200850}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200850	&parmpreflist: user_wants_to_track_thick200850

${METPLUS_USER_WANTS_TO_TRACK_ZETA850}

METplus Config(s)	NML Config File
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA850	&parmpreflist: user_wants_to_track_zeta850

${METPLUS_VERBOSE_VERB}

METplus Config(s)	NML Config File
GFDL_TRACKER_VERBOSE_VERB	&verbose: verb

${METPLUS_VERBOSE_VERB_G2}

METplus Config(s)	NML Config File
GFDL_TRACKER_VERBOSE_VERB_G2	&verbose: verb_g2

6.10 GridDiag

6.10.1 Description

Used to configure the MET tool grid_diag.

6.10.2 METplus Configuration

GRID_DIAG_INPUT_DIR
GRID_DIAG_OUTPUT_DIR
GRID_DIAG_INPUT_TEMPLATE
GRID_DIAG_OUTPUT_TEMPLATE
GRID_DIAG_VERIFICATION_MASK_TEMPLATE
LOG_GRID_DIAG_VERBOSITY
GRID_DIAG_CONFIG_FILE
GRID_DIAG_CUSTOM_LOOP_LIST
GRID_DIAG_INPUT_DATATYPE
GRID_DIAG_REGRID_METHOD
GRID_DIAG_REGRID_WIDTH
GRID_DIAG_REGRID_VLD_THRESH
GRID_DIAG_REGRID_SHAPE
GRID_DIAG_REGRID_TO_GRID
GRID_DIAG_REGRID_CONVERT
GRID_DIAG_REGRID_CENSOR_THRESH
GRID_DIAG_REGRID_CENSOR_VAL
GRID_DIAG_DESC
GRID_DIAG_SKIP_IF_OUTPUT_EXISTS
GRID_DIAG_RUNTIME_FREQ
GRID_DIAG_DESC
GRID_DIAG_MET_CONFIG_OVERRIDES

6.10.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/GridDiagConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Diag configuration file.
//
// For additional information, see the MET_BASE/config/GridDiagConfig_default file.
```

(continues on next page)

(continued from previous page)

```
//
////////////////////////////////////////////////////////////////////////////////

//
// Description
//
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Output grid
//
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
${METPLUS_CENSOR_THRESH}
${METPLUS_CENSOR_VAL}

//
// Data fields
//
${METPLUS_DATA_DICT}

${METPLUS_MASK_DICT}

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_DESC}

METplus Config(s)	MET Config File
DESC or *GRID_DIAG_DESC*	desc

${METPLUS_REGRID_DICT}

METplus Config(s)	MET Config File
GRID_DIAG_REGRID_SHAPE	regrid.shape
GRID_DIAG_REGRID_METHOD	regrid.method
GRID_DIAG_REGRID_WIDTH	regrid.width
GRID_DIAG_REGRID_VLD_THRESH	regrid.vld_thresh
GRID_DIAG_REGRID_TO_GRID	regrid.to_grid
GRID_DIAG_REGRID_CONVERT	regrid.convert
GRID_DIAG_REGRID_CENSOR_THRESH	regrid.censor_thresh
GRID_DIAG_REGRID_CENSOR_VAL	regrid.censor_val

${METPLUS_CENSOR_THRESH}

METplus Config(s)	MET Config File
GRID_DIAG_CENSOR_THRESH	censor_thresh

${METPLUS_CENSOR_VAL}

METplus Config(s)	MET Config File
GRID_DIAG_CENSOR_VAL	censor_val

${METPLUS_DATA_DICT}

METplus Config(s)	MET Config File
BOTH_VAR<n>_NAME	data.field.name
BOTH_VAR<n>_LEVELS	data.field.level
BOTH_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_MASK_DICT}

METplus Config(s)	MET Config File
GRID_DIAG_MASK_GRID	mask.grid
GRID_DIAG_MASK_POLY	mask.poly

Note: Since the default value in the MET config file for 'grid' is grid = ["FULL"];, setting GRID_DIAG_MASK_GRID to an empty string will result in a value of grid = []; in the MET config file.

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
GRID_DIAG_MET_CONFIG_OVERRIDES	n/a

6.11 GridStat

6.11.1 Description

Used to configure the MET tool grid_stat.

6.11.2 METplus Configuration

FCST_GRID_STAT_INPUT_DIR
OBS_GRID_STAT_INPUT_DIR
GRID_STAT_OUTPUT_DIR
FCST_GRID_STAT_INPUT_TEMPLATE
OBS_GRID_STAT_INPUT_TEMPLATE
GRID_STAT_OUTPUT_TEMPLATE
GRID_STAT_VERIFICATION_MASK_TEMPLATE
LOG_GRID_STAT_VERBOSITY
GRID_STAT_OUTPUT_PREFIX
GRID_STAT_CONFIG_FILE
FCST_GRID_STAT_INPUT_DATATYPE
OBS_GRID_STAT_INPUT_DATATYPE
GRID_STAT_ONCE_PER_FIELD
GRID_STAT_CUSTOM_LOOP_LIST
GRID_STAT_SKIP_IF_OUTPUT_EXISTS
GRID_STAT_DESC
GRID_STAT_REGRID_TO_GRID
GRID_STAT_REGRID_METHOD
GRID_STAT_REGRID_WIDTH
GRID_STAT_REGRID_VLD_THRESH
GRID_STAT_REGRID_SHAPE
GRID_STAT_REGRID_CONVERT
GRID_STAT_REGRID_CENSOR_THRESH
GRID_STAT_REGRID_CENSOR_VAL
GRID_STAT_CLIMO_CDF_BINS
GRID_STAT_CLIMO_CDF_CENTER_BINS
GRID_STAT_CLIMO_CDF_WRITE_BINS
GRID_STAT_CLIMO_CDF_DIRECT_PROB
GRID_STAT_OUTPUT_FLAG_FHO

GRID_STAT_OUTPUT_FLAG_CTC
GRID_STAT_OUTPUT_FLAG_CTS
GRID_STAT_OUTPUT_FLAG_MCTC
GRID_STAT_OUTPUT_FLAG_MCTS
GRID_STAT_OUTPUT_FLAG_CNT
GRID_STAT_OUTPUT_FLAG_SL1L2
GRID_STAT_OUTPUT_FLAG_SAL1L2
GRID_STAT_OUTPUT_FLAG_VL1L2
GRID_STAT_OUTPUT_FLAG_VAL1L2
GRID_STAT_OUTPUT_FLAG_VCNT
GRID_STAT_OUTPUT_FLAG_PCT
GRID_STAT_OUTPUT_FLAG_PSTD
GRID_STAT_OUTPUT_FLAG_PJC
GRID_STAT_OUTPUT_FLAG_PRC
GRID_STAT_OUTPUT_FLAG_ECLV
GRID_STAT_OUTPUT_FLAG_NBRCTC
GRID_STAT_OUTPUT_FLAG_NBRCTS
GRID_STAT_OUTPUT_FLAG_NBRCNT
GRID_STAT_OUTPUT_FLAG_GRAD
GRID_STAT_OUTPUT_FLAG_DMAP
GRID_STAT_OUTPUT_FLAG_SEEPS
GRID_STAT_NC_PAIRS_FLAG_LATLON
GRID_STAT_NC_PAIRS_FLAG_RAW
GRID_STAT_NC_PAIRS_FLAG_DIFF
GRID_STAT_NC_PAIRS_FLAG_CLIMO
GRID_STAT_NC_PAIRS_FLAG_CLIMO_CDP
GRID_STAT_NC_PAIRS_FLAG_WEIGHT
GRID_STAT_NC_PAIRS_FLAG_NBRHD
GRID_STAT_NC_PAIRS_FLAG_FOURIER
GRID_STAT_NC_PAIRS_FLAG_GRADIENT
GRID_STAT_NC_PAIRS_FLAG_DISTANCE_MAP
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK
GRID_STAT_NC_PAIRS_FLAG_SEEPS
GRID_STAT_INTERP_FIELD
GRID_STAT_INTERP_VLD_THRESH
GRID_STAT_INTERP_SHAPE
GRID_STAT_INTERP_TYPE_METHOD
GRID_STAT_INTERP_TYPE_WIDTH
GRID_STAT_NC_PAIRS_VAR_NAME
GRID_STAT_GRID_WEIGHT_FLAG
FCST_GRID_STAT_FILE_TYPE

OBS_GRID_STAT_FILE_TYPE
GRID_STAT_CLIMO_MEAN_FILE_NAME
GRID_STAT_CLIMO_MEAN_VAR<n>_NAME
GRID_STAT_CLIMO_MEAN_VAR<n>_LEVELS
GRID_STAT_CLIMO_MEAN_VAR<n>_OPTIONS
GRID_STAT_CLIMO_MEAN_FIELD
GRID_STAT_CLIMO_MEAN_REGRID_METHOD
GRID_STAT_CLIMO_MEAN_REGRID_WIDTH
GRID_STAT_CLIMO_MEAN_REGRID_VLD_THRESH
GRID_STAT_CLIMO_MEAN_REGRID_SHAPE
GRID_STAT_CLIMO_MEAN_TIME_INTERP_METHOD
GRID_STAT_CLIMO_MEAN_MATCH_MONTH
GRID_STAT_CLIMO_MEAN_DAY_INTERVAL
GRID_STAT_CLIMO_MEAN_HOUR_INTERVAL
GRID_STAT_CLIMO_MEAN_USE_FCST
GRID_STAT_CLIMO_MEAN_USE_OBS
GRID_STAT_CLIMO_STDEV_FILE_NAME
GRID_STAT_CLIMO_STDEV_VAR<n>_NAME
GRID_STAT_CLIMO_STDEV_VAR<n>_LEVELS
GRID_STAT_CLIMO_STDEV_VAR<n>_OPTIONS
GRID_STAT_CLIMO_STDEV_FIELD
GRID_STAT_CLIMO_STDEV_REGRID_METHOD
GRID_STAT_CLIMO_STDEV_REGRID_WIDTH
GRID_STAT_CLIMO_STDEV_REGRID_VLD_THRESH
GRID_STAT_CLIMO_STDEV_REGRID_SHAPE
GRID_STAT_CLIMO_STDEV_TIME_INTERP_METHOD
GRID_STAT_CLIMO_STDEV_MATCH_MONTH
GRID_STAT_CLIMO_STDEV_DAY_INTERVAL
GRID_STAT_CLIMO_STDEV_HOUR_INTERVAL
GRID_STAT_CLIMO_STDEV_USE_FCST
GRID_STAT_CLIMO_STDEV_USE_OBS
GRID_STAT_HSS_EC_VALUE
GRID_STAT_DISTANCE_MAP_BADDELEY_P
GRID_STAT_DISTANCE_MAP_BADDELEY_MAX_DIST
GRID_STAT_DISTANCE_MAP_FOM_ALPHA
GRID_STAT_DISTANCE_MAP_ZHU_WEIGHT
GRID_STAT_DISTANCE_MAP_BETA_VALUE_N
GRID_STAT_FOURIER_WAVE_1D_BEG
GRID_STAT_FOURIER_WAVE_1D_END
GRID_STAT_CENSOR_THRESH
GRID_STAT_CENSOR_VAL

FCST_GRID_STAT_IS_PROB
FCST_GRID_STAT_PROB_IN_GRIB_PDS
GRID_STAT_MASK_GRID
GRID_STAT_MASK_POLY
GRID_STAT_MET_CONFIG_OVERRIDES
FCST_GRID_STAT_PROB_THRESH
OBS_GRID_STAT_PROB_THRESH
GRID_STAT_NEIGHBORHOOD_WIDTH
GRID_STAT_NEIGHBORHOOD_SHAPE
GRID_STAT_NEIGHBORHOOD_COV_THRESH
FCST_GRID_STAT_WINDOW_BEGIN
FCST_GRID_STAT_WINDOW_END
OBS_GRID_STAT_WINDOW_BEGIN
OBS_GRID_STAT_WINDOW_END
FCST_GRID_STAT_FILE_WINDOW_BEGIN
FCST_GRID_STAT_FILE_WINDOW_END
OBS_GRID_STAT_FILE_WINDOW_BEGIN
OBS_GRID_STAT_FILE_WINDOW_END
FCST_GRID_STAT_VAR<n>_NAME
FCST_GRID_STAT_VAR<n>_LEVELS
FCST_GRID_STAT_VAR<n>_THRESH
FCST_GRID_STAT_VAR<n>_OPTIONS
OBS_GRID_STAT_VAR<n>_NAME
OBS_GRID_STAT_VAR<n>_LEVELS
OBS_GRID_STAT_VAR<n>_THRESH
OBS_GRID_STAT_VAR<n>_OPTIONS
GRID_STAT_SEEPS_P1_THRESH

Warning: DEPRECATED

GRID_STAT_OUT_DIR
GRID_STAT_CONFIG
CLIMO_GRID_STAT_INPUT_DIR
CLIMO_GRID_STAT_INPUT_TEMPLATE
GRID_STAT_CLIMO_MEAN_INPUT_DIR
GRID_STAT_CLIMO_STDEV_INPUT_DIR
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE

6.11.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/GridStatConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
```

(continues on next page)

(continued from previous page)

```
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
```

(continues on next page)

(continued from previous page)

```
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_DESC}

METplus Config(s)	MET Config File
DESC or *GRID_STAT_DESC*	desc

${METPLUS_OBTYPE}

METplus Config(s)	MET Config File
OBTYPE	obtype

${METPLUS_REGRID_DICT}

METplus Config(s)	MET Config File
GRID_STAT_REGRID_SHAPE	regrid.shape
GRID_STAT_REGRID_METHOD	regrid.method
GRID_STAT_REGRID_WIDTH	regrid.width
GRID_STAT_REGRID_VLD_THRESH	regrid.vld_thresh
GRID_STAT_REGRID_TO_GRID	regrid.to_grid
GRID_STAT_REGRID_CONVERT	regrid.convert
GRID_STAT_REGRID_CENSOR_THRESH	regrid.censor_thresh
GRID_STAT_REGRID_CENSOR_VAL	regrid.censor_val

${METPLUS_FCST_FIELD}

METplus Config(s)	MET Config File
FCST_VAR<n>_NAME	fcst.field.name
FCST_VAR<n>_LEVELS	fcst.field.level
FCST_VAR<n>_THRESH	fcst.field.cat_thresh
FCST_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the forecast field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_FCST_FILE_TYPE}

METplus Config(s)	MET Config File
FCST_GRID_STAT_FILE_TYPE	fcst.file_type

${METPLUS_OBS_FIELD}

METplus Config(s)	MET Config File
OBS_VAR<n>_NAME	fcst.field.name
OBS_VAR<n>_LEVELS	fcst.field.level
OBS_VAR<n>_THRESH	fcst.field.cat_thresh
OBS_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the observation field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_OBS_FILE_TYPE}

METplus Config(s)	MET Config File
OBS_GRID_STAT_FILE_TYPE	obs.file_type

${METPLUS_CLIMO_MEAN_DICT}

METplus Config(s)	MET Config File
GRID_STAT_CLIMO_MEAN_FILE_NAME	climo_mean.file_name
GRID_STAT_CLIMO_MEAN_FIELD	climo_mean.field
GRID_STAT_CLIMO_MEAN_REGRID_METHOD	climo_mean.regrid.method
GRID_STAT_CLIMO_MEAN_REGRID_WIDTH	climo_mean.regrid.width
GRID_STAT_CLIMO_MEAN_REGRID_VLD_THRESH	climo_mean.regrid.vld_thresh
GRID_STAT_CLIMO_MEAN_REGRID_SHAPE	climo_mean.regrid.shape
GRID_STAT_CLIMO_MEAN_TIME_INTERP_METHOD	climo_mean.time_interp_method
GRID_STAT_CLIMO_MEAN_MATCH_MONTH	climo_mean.match_month
GRID_STAT_CLIMO_MEAN_DAY_INTERVAL	climo_mean.day_interval
GRID_STAT_CLIMO_MEAN_HOUR_INTERVAL	climo_mean.hour_interval

${METPLUS_CLIMO_STDEV_DICT}

METplus Config(s)	MET Config File
GRID_STAT_CLIMO_STDEV_FILE_NAME	climo_stdev.file_name
GRID_STAT_CLIMO_STDEV_FIELD	climo_stdev.field
GRID_STAT_CLIMO_STDEV_REGRID_METHOD	climo_stdev.regrid.method
GRID_STAT_CLIMO_STDEV_REGRID_WIDTH	climo_stdev.regrid.width
GRID_STAT_CLIMO_STDEV_REGRID_VLD_THRESH	climo_stdev.regrid.vld_thresh
GRID_STAT_CLIMO_STDEV_REGRID_SHAPE	climo_stdev.regrid.shape
GRID_STAT_CLIMO_STDEV_TIME_INTERP_METHOD	climo_stdev.time_interp_method
GRID_STAT_CLIMO_STDEV_MATCH_MONTH	climo_stdev.match_month
GRID_STAT_CLIMO_STDEV_DAY_INTERVAL	climo_stdev.day_interval
GRID_STAT_CLIMO_STDEV_HOUR_INTERVAL	climo_stdev.hour_interval

${METPLUS_MASK_DICT}

METplus Config(s)	MET Config File
GRID_STAT_MASK_GRID	mask.grid
GRID_STAT_MASK_POLY	mask.poly

Note: Since the default value in the MET config file for 'grid' is grid = ["FULL"];, setting GRID_STAT_MASK_GRID to an empty string will result in a value of grid = []; in the MET config file.

${METPLUS_NBRHD_SHAPE}

METplus Config(s)	MET Config File
GRID_STAT_NEIGHBORHOOD_SHAPE	nbrhd.shape

${METPLUS_NBRHD_WIDTH}

METplus Config(s)	MET Config File
GRID_STAT_NEIGHBORHOOD_WIDTH	nbrhd.width

${METPLUS_NBRHD_COV_THRESH}

METplus Config(s)	MET Config File
GRID_STAT_NEIGHBORHOOD_COV_THRESH	nbrhd.cov_thresh

${METPLUS_OUTPUT_PREFIX}

METplus Config(s)	MET Config File
GRID_STAT_OUTPUT_PREFIX	output_prefix

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
GRID_STAT_MET_CONFIG_OVERRIDES	n/a

${METPLUS_CLIMO_CDF_DICT}

METplus Config(s)	MET Config File
GRID_STAT_CLIMO_CDF_BINS	climo_cdf.cdf_bins
GRID_STAT_CLIMO_CDF_CENTER_BINS	climo_cdf.center_bins
GRID_STAT_CLIMO_CDF_WRITE_BINS	climo_cdf.write_bins
GRID_STAT_CLIMO_CDF_DIRECT_PROB	climo_cdf.direct_prob

${METPLUS_OUTPUT_FLAG_DICT}

METplus Config(s)	MET Config File
GRID_STAT_OUTPUT_FLAG_FHO	output_flag.fho
GRID_STAT_OUTPUT_FLAG_CTC	output_flag.ctc
GRID_STAT_OUTPUT_FLAG_CTS	output_flag.cts
GRID_STAT_OUTPUT_FLAG_MCTC	output_flag.mctc
GRID_STAT_OUTPUT_FLAG_MCTS	output_flag.mcts
GRID_STAT_OUTPUT_FLAG_CNT	output_flag.cnt
GRID_STAT_OUTPUT_FLAG_SL1L2	output_flag.sl1l2
GRID_STAT_OUTPUT_FLAG_SAL1L2	output_flag.sal1l2
GRID_STAT_OUTPUT_FLAG_VL1L2	output_flag.vl1l2
GRID_STAT_OUTPUT_FLAG_VAL1L2	output_flag.val1l2
GRID_STAT_OUTPUT_FLAG_VCNT	output_flag.vcnt
GRID_STAT_OUTPUT_FLAG_PCT	output_flag.pct
GRID_STAT_OUTPUT_FLAG_PSTD	output_flag.pstd
GRID_STAT_OUTPUT_FLAG_PJC	output_flag.pjc
GRID_STAT_OUTPUT_FLAG_PRC	output_flag.prc
GRID_STAT_OUTPUT_FLAG_ECLV	output_flag.eclv
GRID_STAT_OUTPUT_FLAG_NBRCTC	output_flag.nbrctc
GRID_STAT_OUTPUT_FLAG_NBRCTS	output_flag.nbrcts
GRID_STAT_OUTPUT_FLAG_NBRCNT	output_flag.nbrcnt
GRID_STAT_OUTPUT_FLAG_GRAD	output_flag.grad
GRID_STAT_OUTPUT_FLAG_DMAP	output_flag.dmap
GRID_STAT_OUTPUT_FLAG_SEEPS	output_flag.seeps

${METPLUS_NC_PAIRS_FLAG_DICT}

METplus Config(s)	MET Config File
GRID_STAT_NC_PAIRS_FLAG_LATLON	nc_pairs_flag.latlon
GRID_STAT_NC_PAIRS_FLAG_RAW	nc_pairs_flag.raw
GRID_STAT_NC_PAIRS_FLAG_DIFF	nc_pairs_flag.diff
GRID_STAT_NC_PAIRS_FLAG_CLIMO	nc_pairs_flag.climo
GRID_STAT_NC_PAIRS_FLAG_CLIMO_CDP	nc_pairs_flag.climo_cdp
GRID_STAT_NC_PAIRS_FLAG_WEIGHT	nc_pairs_flag.weight
GRID_STAT_NC_PAIRS_FLAG_NBRHD	nc_pairs_flag.nbrhd
GRID_STAT_NC_PAIRS_FLAG_FOURIER	nc_pairs_flag.fourier
GRID_STAT_NC_PAIRS_FLAG_GRADIENT	nc_pairs_flag.gradient
GRID_STAT_NC_PAIRS_FLAG_DISTANCE_MAP	nc_pairs_flag.distance_map
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK	nc_pairs_flag.apply_mask
GRID_STAT_NC_PAIRS_FLAG_SEEPS	nc_pairs_flag.seeps

${METPLUS_INTERP_DICT}

METplus Config(s)	MET Config File
GRID_STAT_INTERP_FIELD	interp.field
GRID_STAT_INTERP_VLD_THRESH	interp.vld_thresh
GRID_STAT_INTERP_SHAPE	interp.shape
GRID_STAT_INTERP_TYPE_METHOD	interp.type.method
GRID_STAT_INTERP_TYPE_WIDTH	interp.type.width

${METPLUS_NC_PAIRS_VAR_NAME}

METplus Config(s)	MET Config File
GRID_STAT_NC_PAIRS_VAR_NAME	nc_pairs_var_name

${METPLUS_GRID_WEIGHT_FLAG}

METplus Config(s)	MET Config File
GRID_STAT_GRID_WEIGHT_FLAG	grid_weight_flag

${METPLUS_HSS_EC_VALUE}

METplus Config(s)	MET Config File
GRID_STAT_HSS_EC_VALUE	hss_ec_value

${METPLUS_DISTANCE_MAP_DICT}

METplus Config(s)	MET Config File
GRID_STAT_DISTANCE_MAP_BADDELEY_P	distance_map.baddeley_p
GRID_STAT_DISTANCE_MAP_BADDELEY_MAX_DIST	distance_map.baddeley_max_dist
GRID_STAT_DISTANCE_MAP_FOM_ALPHA	distance_map.fom_alpha
GRID_STAT_DISTANCE_MAP_ZHU_WEIGHT	distance_map.zhu_weight
GRID_STAT_DISTANCE_MAP_BETA_VALUE_N	distance_map.beta_value(n)

${METPLUS_FOURIER_DICT}

METplus Config(s)	MET Config File
GRID_STAT_FOURIER_WAVE_1D_BEG	fourier.wave_1d_beg
GRID_STAT_FOURIER_WAVE_1D_END	fourier.wave_1d_end

${METPLUS_CENSOR_THRESH}

METplus Config(s)	MET Config File
GRID_STAT_CENSOR_THRESH	censor_thresh

${METPLUS_CENSOR_VAL}

METplus Config(s)	MET Config File
GRID_STAT_CENSOR_VAL	censor_val

${METPLUS_SEEPS_P1_THRESH}

METplus Config(s)	MET Config File
GRID_STAT_SEEPS_P1_THRESH	seeps_p1_thresh

6.12 IODA2NC

6.12.1 Description

Used to configure the MET tool ioda2nc

6.12.2 METplus Configuration

IODA2NC_INPUT_DIR
IODA2NC_INPUT_TEMPLATE
IODA2NC_OUTPUT_DIR
IODA2NC_OUTPUT_TEMPLATE
LOG_IODA2NC_VERBOSITY
IODA2NC_SKIP_IF_OUTPUT_EXISTS
IODA2NC_CONFIG_FILE
IODA2NC_FILE_WINDOW_BEG
IODA2NC_FILE_WINDOW_END
IODA2NC_VALID_BEG
IODA2NC_VALID_END
IODA2NC_NMSG
IODA2NC_MESSAGE_TYPE
IODA2NC_MESSAGE_TYPE_MAP
IODA2NC_MESSAGE_TYPE_GROUP_MAP
IODA2NC_STATION_ID
IODA2NC_OBS_WINDOW_BEG
IODA2NC_OBS_WINDOW_END
IODA2NC_MASK_GRID
IODA2NC_MASK_POLY
IODA2NC_ELEVATION_RANGE_BEG
IODA2NC_ELEVATION_RANGE_END
IODA2NC_LEVEL_RANGE_BEG
IODA2NC_LEVEL_RANGE_END

IODA2NC_OBS_VAR
IODA2NC_OBS_NAME_MAP
IODA2NC_METADATA_MAP
IODA2NC_MISSING_THRESH
IODA2NC_QUALITY_MARK_THRESH
IODA2NC_TIME_SUMMARY_FLAG
IODA2NC_TIME_SUMMARY_RAW_DATA
IODA2NC_TIME_SUMMARY_BEG
IODA2NC_TIME_SUMMARY_END
IODA2NC_TIME_SUMMARY_STEP
IODA2NC_TIME_SUMMARY_WIDTH
IODA2NC_TIME_SUMMARY_GRIB_CODE
IODA2NC_TIME_SUMMARY_OBS_VAR
IODA2NC_TIME_SUMMARY_TYPE
IODA2NC_TIME_SUMMARY_VLD_FREQ
IODA2NC_TIME_SUMMARY_VLD_THRESH
IODA2NC_CUSTOM_LOOP_LIST
IODA2NC_MET_CONFIG_OVERRIDES

6.12.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/IODA2NCConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// IODA2NC configuration file.
//
// For additional information, please see the MET Users Guide.
//
////////////////////////////////////////////////////////////////////////////////

//
// IODA message type
//
// message_type = [
```

(continues on next page)

(continued from previous page)

```
${METPLUS_MESSAGE_TYPE}

//
// Mapping of message type group name to comma-separated list of values
// Derive PRMSL only for SURFACE message types
//
// message_type_group_map = [
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

//
// Mapping of input IODA message types to output message types
//
// message_type_map = [
${METPLUS_MESSAGE_TYPE_MAP}

//
// IODA station ID
//
// station_id = [
${METPLUS_STATION_ID}

////////////////////////////////////////////////////////////////////////////////

//
// Observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observation retention regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observing location elevation
//
// elevation_range = {
${METPLUS_ELEVATION_RANGE_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Vertical levels to retain
//
// level_range = {
${METPLUS_LEVEL_RANGE_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// IODA variable names to retain or derive.
// Use obs_bufr_map to rename variables in the output.
// If empty or 'all', process all available variables.
//
// obs_var = [
${METPLUS_OBS_VAR}

////////////////////////////////////////////////////////////////////////////////

//
// Mapping of input IODA variable names to output variables names.
// The default IODA map, obs_var_map, is appended to this map.
//
// obs_name_map = [
${METPLUS_OBS_NAME_MAP}

//
// Default mapping for Metadata.
//
// metadata_map = [
${METPLUS_METADATA_MAP}

// missing_thresh = [
${METPLUS_MISSING_THRESH}

////////////////////////////////////////////////////////////////////////////////

// quality_mark_thresh =
${METPLUS_QUALITY_MARK_THRESH}

////////////////////////////////////////////////////////////////////////////////

//
// Time periods for the summarization
```

(continues on next page)

(continued from previous page)

```
// obs_var (string array) is added and works like grib_code (int array)
// when use_var_id is enabled and variable names are saved.
//
// time_summary = {
${METPLUS_TIME_SUMMARY_DICT}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

//version = "V10.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MESSAGE_TYPE}

METplus Config(s)	MET Config File
IODA2NC_MESSAGE_TYPE	message_type

${METPLUS_MESSAGE_TYPE_MAP}

METplus Config(s)	MET Config File
IODA2NC_MESSAGE_TYPE_MAP	message_type_map

${METPLUS_MESSAGE_TYPE_GROUP_MAP}

METplus Config(s)	MET Config File
IODA2NC_MESSAGE_TYPE_GROUP_MAP	message_type_group_map

${METPLUS_STATION_ID}

METplus Config(s)	MET Config File
IODA2NC_STATION_ID	station_id

${METPLUS_OBS_WINDOW_DICT}

METplus Config(s)	MET Config File
IODA2NC_OBS_WINDOW_BEG	obs_window.beg
IODA2NC_OBS_WINDOW_END	obs_window.end

${METPLUS_MASK_DICT}

METplus Config(s)	MET Config File
IODA2NC_MASK_GRID	mask.grid
IODA2NC_MASK_POLY	mask.poly

${METPLUS_ELEVATION_RANGE_DICT}

METplus Config(s)	MET Config File
IODA2NC_ELEVATION_RANGE_BEG	elevation_range.beg
IODA2NC_ELEVATION_RANGE_END	elevation_range.end

${METPLUS_LEVEL_RANGE_DICT}

METplus Config(s)	MET Config File
IODA2NC_LEVEL_RANGE_BEG	level_range.beg
IODA2NC_LEVEL_RANGE_END	level_range.end

${METPLUS_OBS_VAR}

METplus Config(s)	MET Config File
IODA2NC_OBS_VAR	obs_var

${METPLUS_OBS_NAME_MAP}

METplus Config(s)	MET Config File
IODA2NC_OBS_NAME_MAP	obs_name_map

${METPLUS_METADATA_MAP}

METplus Config(s)	MET Config File
IODA2NC_METADATA_MAP	metadata_map

${METPLUS_MISSING_THRESH}

METplus Config(s)	MET Config File
IODA2NC_MISSING_THRESH	missing_thresh

${METPLUS_QUALITY_MARK_THRESH}

METplus Config(s)	MET Config File
IODA2NC_QUALITY_MARK_THRESH	quality_mark_thresh

${METPLUS_TIME_SUMMARY_DICT}

METplus Config(s)	MET Config File
IODA2NC_TIME_SUMMARY_FLAG	time_summary.flag
IODA2NC_TIME_SUMMARY_RAW_DATA	time_summary.raw_data
IODA2NC_TIME_SUMMARY_BEG	time_summary.beg
IODA2NC_TIME_SUMMARY_END	time_summary.end
IODA2NC_TIME_SUMMARY_STEP	time_summary.step
IODA2NC_TIME_SUMMARY_WIDTH	time_summary.width
IODA2NC_TIME_SUMMARY_GRIB_CODE	time_summary.grib_code
IODA2NC_TIME_SUMMARY_OBS_VAR	time_summary.obs_var
IODA2NC_TIME_SUMMARY_TYPE	time_summary.type
IODA2NC_TIME_SUMMARY_VLD_FREQ	time_summary.vld_freq
IODA2NC_TIME_SUMMARY_VLD_THRESH	time_summary.vld_thresh

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
IODA2NC_MET_CONFIG_OVERRIDES	n/a

6.13 METdbLoad

6.13.1 Description

Used to call the met_db_load.py script from dtcenter/METdataio to load MET output into a METviewer database.

6.13.2 METplus Configuration

MET_DB_LOAD_RUNTIME_FREQ
MET_DATA_DB_DIR
MET_DB_LOAD_XML_FILE
MET_DB_LOAD_REMOVE_TMP_XML
MET_DB_LOAD_MV_HOST
MET_DB_LOAD_MV_DATABASE
MET_DB_LOAD_MV_USER
MET_DB_LOAD_MV_PASSWORD
MET_DB_LOAD_MV_VERBOSE
MET_DB_LOAD_MV_INSERT_SIZE
MET_DB_LOAD_MV_MODE_HEADER_DB_CHECK
MET_DB_LOAD_MV_DROP_INDEXES
MET_DB_LOAD_MV_APPLY_INDEXES
MET_DB_LOAD_MV_GROUP

MET_DB_LOAD_MV_LOAD_STAT
MET_DB_LOAD_MV_LOAD_MODE
MET_DB_LOAD_MV_LOAD_MTD
MET_DB_LOAD_MV_LOAD_MPR
MET_DB_LOAD_INPUT_TEMPLATE

6.13.3 XML Configuration

Below is the XML template configuration file used for this wrapper. The wrapper substitutes values from the METplus configuration file into this configuration file. While it may appear that environment variables are used in the XML template file, they are not actually environment variables. The wrapper searches for these strings and substitutes the values as appropriate.

```
<load_spec>
  <connection>
    <host>${METPLUS_MV_HOST}</host>
    <database>${METPLUS_MV_DATABASE}</database>
    <user>${METPLUS_MV_USER}</user>
    <password>${METPLUS_MV_PASSWORD}</password>
  </connection>

  <verbose>${METPLUS_MV_VERBOSE}</verbose>
  <insert_size>${METPLUS_MV_INSERT_SIZE}</insert_size>
  <mode_header_db_check>${METPLUS_MV_MODE_HEADER_DB_CHECK}</mode_header_db_check>
  <drop_indexes>${METPLUS_MV_DROP_INDEXES}</drop_indexes>
  <apply_indexes>${METPLUS_MV_APPLY_INDEXES}</apply_indexes>
  <group>${METPLUS_MV_GROUP}</group>
  <load_stat>${METPLUS_MV_LOAD_STAT}</load_stat>
  <load_mode>${METPLUS_MV_LOAD_MODE}</load_mode>
  <load_mtd>${METPLUS_MV_LOAD_MTD}</load_mtd>
  <load_mpr>${METPLUS_MV_LOAD_MPR}</load_mpr>

  <folder_tmpl>{dirs}</folder_tmpl>
  <load_val>
    <field name="dirs">
      ${METPLUS_INPUT_PATHS}
    </field>
  </load_val>
</load_spec>
```

${METPLUS_MV_HOST}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_HOST	<load_spec><connection><host>

${METPLUS_MV_DATABASE}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_DATABASE	<load_spec><connection><database>

${METPLUS_MV_USER}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_USER	<load_spec><connection><user>

${METPLUS_MV_PASSWORD}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_PASSWORD	<load_spec><connection><password>

${METPLUS_MV_VERBOSE}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_VERBOSE	<load_spec><verbose>

${METPLUS_MV_INSERT_SIZE}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_INSERT_SIZE	<load_spec><insert_size>

${METPLUS_MV_MODE_HEADER_DB_CHECK}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_MODE_HEADER_DB_CHECK	<load_spec><mode_header_db_check>

${METPLUS_MV_DROP_INDEXES}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_DROP_INDEXES	<load_spec><drop_indexes>

${METPLUS_MV_APPLY_INDEXES}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_APPLY_INDEXES	<load_spec><apply_indexes>

${METPLUS_MV_GROUP}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_GROUP	<load_spec><group>

${METPLUS_MV_LOAD_STAT}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_LOAD_STAT	<load_spec><load_stat>

${METPLUS_MV_LOAD_MODE}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_LOAD_MODE	<load_spec><load_mode>

${METPLUS_MV_LOAD_MTD}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_LOAD_MTD	<load_spec><load_mtd>

${METPLUS_MV_LOAD_MPR}

METplus Config(s)	XML Config File
MET_DB_LOAD_MV_LOAD_MPR	<load_spec><load_mpr>

${METPLUS_INPUT_PATHS}

METplus Config(s)	XML Config File
MET_DB_LOAD_INPUT_TEMPLATE	<load_val><field name="dirs"><val>

6.14 MODE

6.14.1 Description

Used to configure the MET Method for Object-based Diagnostic Evaluation tool mode.

6.14.2 METplus Configuration

FCST_MODE_INPUT_DIR
OBS_MODE_INPUT_DIR
MODE_OUTPUT_DIR
FCST_MODE_INPUT_TEMPLATE
OBS_MODE_INPUT_TEMPLATE
MODE_OUTPUT_TEMPLATE
MODE_VERIFICATION_MASK_TEMPLATE
LOG_MODE_VERBOSITY
MODE_OUTPUT_PREFIX

MODE_REGRID_TO_GRID
MODE_REGRID_METHOD
MODE_REGRID_WIDTH
MODE_REGRID_VLD_THRESH
MODE_REGRID_SHAPE
MODE_REGRID_CONVERT
MODE_REGRID_CENSOR_THRESH
MODE_REGRID_CENSOR_VAL
MODE_CONFIG_FILE
FCST_MODE_INPUT_DATATYPE
OBS_MODE_INPUT_DATATYPE
MODE_QUILT
MODE_CONV_RADIUS
FCST_MODE_CONV_RADIUS
OBS_MODE_CONV_RADIUS
MODE_CONV_THRESH
FCST_MODE_CONV_THRESH
OBS_MODE_CONV_THRESH
MODE_MERGE_THRESH
FCST_MODE_MERGE_THRESH
OBS_MODE_MERGE_THRESH
MODE_MERGE_FLAG
FCST_MODE_MERGE_FLAG
OBS_MODE_MERGE_FLAG
MODE_MERGE_CONFIG_FILE
FCST_MODE_WINDOW_BEGIN
FCST_MODE_WINDOW_END
OBS_MODE_WINDOW_BEGIN
OBS_MODE_WINDOW_END
FCST_MODE_FILE_WINDOW_BEGIN
FCST_MODE_FILE_WINDOW_END
OBS_MODE_FILE_WINDOW_BEGIN
OBS_MODE_FILE_WINDOW_END
MODE_CUSTOM_LOOP_LIST
MODE_SKIP_IF_OUTPUT_EXISTS
MODE_DESC
MODE_MET_CONFIG_OVERRIDES
MODE_WEIGHT_CENTROID_DIST
MODE_WEIGHT_BOUNDARY_DIST
MODE_WEIGHT_CONVEX_HULL_DIST
MODE_WEIGHT_ANGLE_DIFF

FCST_MODE_VAR<n>_LEVELS
FCST_MODE_VAR<n>_THRESH
FCST_MODE_VAR<n>_OPTIONS
MODE_FCST_FILE_TYPE
MODE_FCST_MULTIVAR_NAME
MODE_FCST_MULTIVAR_LEVEL
OBS_MODE_VAR<n>_NAME
OBS_MODE_VAR<n>_LEVELS
OBS_MODE_VAR<n>_THRESH
OBS_MODE_VAR<n>_OPTIONS
MODE_OBS_FILE_TYPE
MODE_OBS_MULTIVAR_NAME
MODE_OBS_MULTIVAR_LEVEL

Warning: DEPRECATED:

MODE_OUT_DIR
MODE_CONFIG

6.14.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/MODEConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Approximate grid resolution (km)
//
// grid_res =
${METPLUS_GRID_RES}

////////////////////////////////////////////////////////////////////////////////

//
// Run all permutations of radius and threshold
//
// quilt =
${METPLUS_QUILT}

//
// MODE Multivar boolean combination logic
```

(continues on next page)

(continued from previous page)

```
//
//multivar_logic =
${METPLUS_MULTIVAR_LOGIC}

//multivar_intensity_flag =
${METPLUS_MULTIVAR_INTENSITY_FLAG}

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FIELD}

   ${METPLUS_FCST_CENSOR_THRESH}
   ${METPLUS_FCST_CENSOR_VAL}
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}
   ${METPLUS_FCST_VLD_THRESH}
   ${METPLUS_FCST_FILTER_ATTR_NAME}
   ${METPLUS_FCST_FILTER_ATTR_THRESH}
   ${METPLUS_FCST_MERGE_THRESH}
   ${METPLUS_FCST_MERGE_FLAG}
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_MULTIVAR_NAME}
   ${METPLUS_FCST_MULTIVAR_LEVEL}
}

obs = {
   ${METPLUS_OBS_FIELD}

   ${METPLUS_OBS_CENSOR_THRESH}
   ${METPLUS_OBS_CENSOR_VAL}
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
   ${METPLUS_OBS_VLD_THRESH}
   ${METPLUS_OBS_FILTER_ATTR_NAME}
   ${METPLUS_OBS_FILTER_ATTR_THRESH}
   ${METPLUS_OBS_MERGE_THRESH}
   ${METPLUS_OBS_MERGE_FLAG}
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_MULTIVAR_NAME}
   ${METPLUS_OBS_MULTIVAR_LEVEL}
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Handle missing data
//
// mask_missing_flag =
${METPLUS_MASK_MISSING_FLAG}

//
// Match objects between the forecast and observation fields
//
//match_flag =
${METPLUS_MATCH_FLAG}

//
// Maximum centroid distance for objects to be compared
//
//max_centroid_dist =
${METPLUS_MAX_CENTROID_DIST}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine weights
//
//weight = {
${METPLUS_WEIGHT_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine interest functions
//
interest_function = {

   ${METPLUS_INTEREST_FUNCTION_CENTROID_DIST}

   ${METPLUS_INTEREST_FUNCTION_BOUNDARY_DIST}
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_INTEREST_FUNCTION_CONVEX_HULL_DIST}

   angle_diff = (
      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 0.0 )
   );

   aspect_diff = (
      (  0.00, 1.0 )
      (  0.10, 1.0 )
      (  0.75, 0.0 )
   );

   corner   = 0.8;
   ratio_if = (
      (    0.0, 0.0 )
      ( corner, 1.0 )
      (    1.0, 1.0 )
   );

   area_ratio = ratio_if;

   int_area_ratio = (
      ( 0.00, 0.00 )
      ( 0.10, 0.50 )
      ( 0.25, 1.00 )
      ( 1.00, 1.00 )
   );

   curvature_ratio = ratio_if;

   complexity_ratio = ratio_if;

   inten_perc_ratio = ratio_if;
}

////////////////////////////////////////////////////////////////////////////////

//
// Total interest threshold for determining matches
//
//total_interest_thresh =
${METPLUS_TOTAL_INTEREST_THRESH}
```

(continues on next page)

(continued from previous page)

```
//
// Interest threshold for printing output pair information
//
print_interest_thresh = 0.0;

////////////////////////////////////////////////////////////////////////////////

//
// Plotting information
//
met_data_dir = "MET_BASE";

fcst_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

obs_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

object_plot = {
   color_table      = "MET_BASE/colortables/mode_obj.ctable";
}

//
// Boolean for plotting on the region of valid data within the domain
//
plot_valid_flag = FALSE;

//
// Plot polyline edges using great circle arcs instead of straight lines
//
plot_gcarc_flag = FALSE;

////////////////////////////////////////////////////////////////////////////////

//
// NetCDF matched pairs, PostScript, and contingency table output files
//
//ps_plot_flag =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_PS_PLOT_FLAG}

//nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

//ct_stats_flag =
${METPLUS_CT_STATS_FLAG}


////////////////////////////////////////////////////////////////////////////////

shift_right = 0;   //  grid squares

////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_DESC}

METplus Config(s)	MET Config File
MODE_DESC	desc

${METPLUS_OBTYPE}

METplus Config(s)	MET Config File
OBTYPE	obtype

${METPLUS_REGRID_DICT}

METplus Config(s)	MET Config File
MODE_REGRID_SHAPE	regrid.shape
MODE_REGRID_METHOD	regrid.method
MODE_REGRID_WIDTH	regrid.width
MODE_REGRID_VLD_THRESH	regrid.vld_thresh
MODE_REGRID_TO_GRID	regrid.to_grid
MODE_REGRID_CONVERT	regrid.convert
MODE_REGRID_CENSOR_THRESH	regrid.censor_thresh
MODE_REGRID_CENSOR_VAL	regrid.censor_val

${METPLUS_GRID_RES}

METplus Config(s)	MET Config File
MODE_GRID_RES	grid_res

${METPLUS_QUILT}

METplus Config(s)	MET Config File
MODE_QUILT	quilt

${METPLUS_MULTIVAR_LOGIC}

METplus Config(s)	MET Config File
MODE_MULTIVAR_LOGIC	multivar_logic

${METPLUS_MULTIVAR_INTENSITY_FLAG}

METplus Config(s)	MET Config File
MODE_MULTIVAR_INTENSITY_FLAG	multivar_intensity_flag

${METPLUS_FCST_FIELD}

METplus Config(s)	MET Config File
FCST_VAR<n>_NAME	fcst.field.name
FCST_VAR<n>_LEVELS	fcst.field.level
FCST_VAR<n>_THRESH	fcst.field.cat_thresh
FCST_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the forecast field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_FCST_CONV_RADIUS}

METplus Config(s)	MET Config File
MODE_FCST_CONV_RADIUS	fcst.conv_radius

${METPLUS_FCST_CONV_THRESH}

METplus Config(s)	MET Config File
MODE_FCST_CONV_THRESH	fcst.conv_thresh

${METPLUS_FCST_MERGE_THRESH}

METplus Config(s)	MET Config File
MODE_FCST_MERGE_THRESH	fcst.merge_thresh

${METPLUS_FCST_MERGE_FLAG}

METplus Config(s)	MET Config File
MODE_FCST_MERGE_FLAG	fcst.merge_flag

${METPLUS_FCST_FILE_TYPE}

METplus Config(s)	MET Config File
MODE_FCST_FILE_TYPE	fcst.file_type

${METPLUS_FCST_MULTIVAR_NAME}

METplus Config(s)	MET Config File
MODE_FCST_MULTIVAR_NAME	fcst.multivar_name

${METPLUS_FCST_MULTIVAR_LEVEL}

METplus Config(s)	MET Config File
MODE_FCST_MULTIVAR_LEVEL	fcst.multivar_level

${METPLUS_OBS_FIELD}

METplus Config(s)	MET Config File
OBS_VAR<n>_NAME	fcst.field.name
OBS_VAR<n>_LEVELS	fcst.field.level
OBS_VAR<n>_THRESH	fcst.field.cat_thresh
OBS_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the observation field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_OBS_CONV_RADIUS}

METplus Config(s)	MET Config File
OBS_MODE_CONV_RADIUS	obs.conv_radius

${METPLUS_OBS_CONV_THRESH}

METplus Config(s)	MET Config File
OBS_MODE_CONV_THRESH	obs.conv_thresh

${METPLUS_OBS_MERGE_THRESH}

METplus Config(s)	MET Config File
OBS_MODE_MERGE_THRESH	obs.merge_thresh

${METPLUS_OBS_MERGE_FLAG}

METplus Config(s)	MET Config File
OBS_MODE_MERGE_FLAG	obs.merge_flag

${METPLUS_OBS_FILE_TYPE}

METplus Config(s)	MET Config File
MODE_OBS_FILE_TYPE	obs.file_type

${METPLUS_OBS_MULTIVAR_NAME}

METplus Config(s)	MET Config File
MODE_OBS_MULTIVAR_NAME	obs.multivar_name

${METPLUS_OBS_MULTIVAR_LEVEL}

METplus Config(s)	MET Config File
MODE_OBS_MULTIVAR_LEVEL	obs.multivar_level

${METPLUS_MASK_POLY}

METplus Config(s)	MET Config File
MODE_MASK_POLY	mask.poly

${METPLUS_OUTPUT_PREFIX}

METplus Config(s)	MET Config File
MODE_OUTPUT_PREFIX	output_prefix

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
MODE_MET_CONFIG_OVERRIDES	n/a

${METPLUS_FCST_FILTER_ATTR_NAME}

METplus Config(s)	MET Config File
MODE_FCST_FILTER_ATTR_NAME	fcst.filter_attr_name

${METPLUS_FCST_FILTER_ATTR_THRESH}

METplus Config(s)	MET Config File
MODE_FCST_FILTER_ATTR_THRESH	fcst.filter_attr_thresh

${METPLUS_FCST_CENSOR_THRESH}

METplus Config(s)	MET Config File
MODE_FCST_CENSOR_THRESH	fcst.censor_thresh

${METPLUS_FCST_CENSOR_VAL}

METplus Config(s)	MET Config File
MODE_FCST_CENSOR_VAL	fcst.censor_val

${METPLUS_FCST_VLD_THRESH}

METplus Config(s)	MET Config File
MODE_FCST_VLD_THRESH	fcst.vld_thresh

${METPLUS_OBS_FILTER_ATTR_NAME}

METplus Config(s)	MET Config File
MODE_OBS_FILTER_ATTR_NAME	obs.filter_attr_name

${METPLUS_OBS_FILTER_ATTR_THRESH}

METplus Config(s)	MET Config File
MODE_OBS_FILTER_ATTR_THRESH	obs.filter_attr_thresh

${METPLUS_OBS_CENSOR_THRESH}

METplus Config(s)	MET Config File
MODE_OBS_CENSOR_THRESH	obs.censor_thresh

${METPLUS_OBS_CENSOR_VAL}

METplus Config(s)	MET Config File
MODE_OBS_CENSOR_VAL	obs.censor_val

${METPLUS_OBS_VLD_THRESH}

METplus Config(s)	MET Config File
MODE_OBS_VLD_THRESH	obs.vld_thresh

${METPLUS_MASK_DICT}

METplus Config(s)	MET Config File
MODE_MASK_GRID	mask.grid
MODE_MASK_GRID_FLAG	mask.grid_flag
MODE_MASK_POLY	mask.poly
MODE_MASK_POLY_FLAG	mask.poly_flag

${METPLUS_MASK_MISSING_FLAG}

METplus Config(s)	MET Config File
MODE_MASK_MISSING_FLAG	mask_missing_flag

${METPLUS_MATCH_FLAG}

METplus Config(s)	MET Config File
MODE_MATCH_FLAG	match_flag

${METPLUS_WEIGHT_DICT}

METplus Config(s)	MET Config File
MODE_WEIGHT_CENTROID_DIST	weight.centroid_dist
MODE_WEIGHT_BOUNDARY_DIST	weight.boundary_dist
MODE_WEIGHT_CONVEX_HULL_DIST	weight.convex_hull_dist
MODE_WEIGHT_ANGLE_DIFF	weight.angle_diff
MODE_WEIGHT_ASPECT_DIFF	weight.aspect_diff
MODE_WEIGHT_AREA_RATIO	weight.area_ratio
MODE_WEIGHT_INT_AREA_RATIO	weight.int_area_ratio
MODE_WEIGHT_CURVATURE_RATIO	weight.curvature_ratio
MODE_WEIGHT_COMPLEXITY_RATIO	weight.complexity_ratio
MODE_WEIGHT_INTEN_PERC_RATIO	weight.inten_perc_ratio
MODE_WEIGHT_INTEN_PERC_VALUE	weight.inten_perc_value

${METPLUS_NC_PAIRS_FLAG_DICT}

METplus Config(s)	MET Config File
MODE_NC_PAIRS_FLAG_LATLON	nc_pairs_flag.latlon
MODE_NC_PAIRS_FLAG_RAW	nc_pairs_flag.raw
MODE_NC_PAIRS_FLAG_OBJECT_RAW	nc_pairs_flag.object_raw
MODE_NC_PAIRS_FLAG_OBJECT_ID	nc_pairs_flag.object_id
MODE_NC_PAIRS_FLAG_CLUSTER_ID	nc_pairs_flag.cluster_id
MODE_NC_PAIRS_FLAG_POLYLINES	nc_pairs_flag.polylines

${METPLUS_MAX_CENTROID_DIST}

METplus Config(s)	MET Config File
MODE_MAX_CENTROID_DIST	max_centroid_dist

${METPLUS_INTEREST_FUNCTION_CENTROID_DIST}

METplus Config(s)	MET Config File
MODE_INTEREST_FUNCTION_CENTROID_DIST	interest_function.centroid_dist

${METPLUS_INTEREST_FUNCTION_BOUNDARY_DIST}

METplus Config(s)	MET Config File
MODE_INTEREST_FUNCTION_BOUNDARY_DIST	interest_function.boundary_dist

${METPLUS_INTEREST_FUNCTION_CONVEX_HULL_DIST}

METplus Config(s)	MET Config File
MODE_INTEREST_FUNCTION_CONVEX_HULL_DIST	interest_function.convex_hull_dist

${METPLUS_TOTAL_INTEREST_THRESH}

METplus Config(s)	MET Config File
MODE_TOTAL_INTEREST_THRESH	total_interest_thresh

${METPLUS_PS_PLOT_FLAG}

METplus Config(s)	MET Config File
MODE_PS_PLOT_FLAG	ps_plot_flag

${METPLUS_CT_STATS_FLAG}

METplus Config(s)	MET Config File
MODE_CT_STATS_FLAG	ct_stats_flag

6.15 MTD

6.15.1 Description

Used to configure the MET MODE Time Domain tool mtd. This tools follows objects through time and can also be used to track objects.

6.15.2 METplus Configuration

FCST_MTD_INPUT_DIR
OBS_MTD_INPUT_DIR
MTD_OUTPUT_DIR
FCST_MTD_INPUT_TEMPLATE
OBS_MTD_INPUT_TEMPLATE
FCST_MTD_INPUT_FILE_LIST
OBS_MTD_INPUT_FILE_LIST
MTD_OUTPUT_TEMPLATE
MTD_CONFIG_FILE
MTD_MIN_VOLUME
MTD_SINGLE_RUN
MTD_SINGLE_DATA_SRC
FCST_MTD_INPUT_DATATYPE
OBS_MTD_INPUT_DATATYPE
FCST_MTD_CONV_RADIUS
FCST_MTD_CONV_THRESH
OBS_MTD_CONV_RADIUS
OBS_MTD_CONV_THRESH
MTD_CUSTOM_LOOP_LIST

MTD_SKIP_IF_OUTPUT_EXISTS
MTD_DESC
MTD_REGRID_TO_GRID
MTD_REGRID_METHOD
MTD_REGRID_WIDTH
MTD_REGRID_VLD_THRESH
MTD_REGRID_SHAPE
MTD_REGRID_CONVERT
MTD_REGRID_CENSOR_THRESH
MTD_REGRID_CENSOR_VAL
MTD_MET_CONFIG_OVERRIDES
FCST_MTD_IS_PROB
FCST_MTD_PROB_IN_GRIB_PDS
FCST_MTD_VAR<n>_NAME
FCST_MTD_VAR<n>_LEVELS
FCST_MTD_VAR<n>_THRESH
FCST_MTD_VAR<n>_OPTIONS
OBS_MTD_VAR<n>_NAME
OBS_MTD_VAR<n>_LEVELS
OBS_MTD_VAR<n>_THRESH
OBS_MTD_VAR<n>_OPTIONS

Warning: DEPRECATED:

MTD_OUT_DIR
MTD_CONFIG
MTD_SINGLE_RUN_SRC

6.15.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/MTDConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed

examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE Time Domain configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

   //
   // Output model name to be written
   //

${METPLUS_MODEL}

   //
   // Output description to be written
   //

${METPLUS_DESC}

   //
   // Output observation type to be written
   //

${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

   //
   //  Verification grid
   //  May be set separately in each "field" entry
   //
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

   //
   // Approximate grid resolution (km)
   //

grid_res = 4;

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
   //
   // Forecast and observation fields to be verified
   //

fcst = {

   ${METPLUS_FCST_FILE_TYPE}

   ${METPLUS_FCST_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}

}

obs = {

   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
}

////////////////////////////////////////////////////////////////////////////////

   //
   // Intensity percentile value to be written
   //

inten_perc_value = 99;

////////////////////////////////////////////////////////////////////////////////

   //
   //  Throw away 3D objects with volumes smaller than this
   //
${METPLUS_MIN_VOLUME}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine weights
   //

weight = {

   space_centroid_dist  = 1.0;

   time_centroid_delta  = 1.0;

   speed_delta          = 1.0;

   direction_diff       = 1.0;

   volume_ratio         = 1.0;

   axis_angle_diff      = 1.0;

   start_time_delta     = 1.0;

   end_time_delta       = 1.0;

}

////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine interest functions
   //

interest_function = {

   space_centroid_dist = (

      (   0.0, 1.0 )
      (  50.0, 0.5 )
      ( 100.0, 0.0 )

   );

   time_centroid_delta = (
```

(continues on next page)

(continued from previous page)

```
   ( -3.0, 0.0 )
   ( -2.0, 0.5 )
   ( -1.0, 0.8 )
   (  0.0, 1.0 )
   (  1.0, 0.8 )
   (  2.0, 0.5 )
   (  3.0, 0.0 )

);

speed_delta = (

   ( -10.0, 0.0 )
   (  -5.0, 0.5 )
   (   0.0, 1.0 )
   (   5.0, 0.5 )
   (  10.0, 0.0 )

);

direction_diff = (

   (   0.0, 1.0 )
   (  90.0, 0.0 )
   ( 180.0, 0.0 )

);

volume_ratio = (

   (  0.0, 0.0 )
   (  0.5, 0.5 )
   (  1.0, 1.0 )
   (  1.5, 0.5 )
   (  2.0, 0.0 )

);

axis_angle_diff = (

   (  0.0, 1.0 )
   ( 30.0, 1.0 )
   ( 90.0, 0.0 )
```

(continues on next page)

(continued from previous page)

```
   );

   start_time_delta = (

      ( -5.0, 0.0 )
      ( -3.0, 0.5 )
      (  0.0, 1.0 )
      (  3.0, 0.5 )
      (  5.0, 0.0 )

   );

   end_time_delta = (

      ( -5.0, 0.0 )
      ( -3.0, 0.5 )
      (  0.0, 1.0 )
      (  3.0, 0.5 )
      (  5.0, 0.0 )

   );

}   //  interest functions


////////////////////////////////////////////////////////////////////////////////

   //
   // Total interest threshold for determining matches
   //

total_interest_thresh = 0.7;


////////////////////////////////////////////////////////////////////////////////

   //
   // Output flags
   //

nc_output = {

   latlon       = true;
   raw          = true;
   object_id    = true;
```

(continues on next page)

(continued from previous page)

```
   cluster_id   = true;

}

txt_output = {

   attributes_2d   = true;
   attributes_3d   = true;

}


////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_DESC}

METplus Config(s)	MET Config File
MTD_DESC	desc

${METPLUS_OBTYPE}

METplus Config(s)	MET Config File
OBTYPE	obtype

${METPLUS_REGRID_DICT}

METplus Config(s)	MET Config File
MTD_REGRID_SHAPE	regrid.shape
MTD_REGRID_METHOD	regrid.method
MTD_REGRID_WIDTH	regrid.width
MTD_REGRID_VLD_THRESH	regrid.vld_thresh
MTD_REGRID_TO_GRID	regrid.to_grid
MTD_REGRID_CONVERT	regrid.convert
MTD_REGRID_CENSOR_THRESH	regrid.censor_thresh
MTD_REGRID_CENSOR_VAL	regrid.censor_val

${METPLUS_FCST_FILE_TYPE}

METplus Config(s)	MET Config File
FCST_MTD_INPUT_DATATYPE	fcst.file_type

${METPLUS_FCST_FIELD}

METplus Config(s)	MET Config File
FCST_VAR<n>_NAME	fcst.field.name
FCST_VAR<n>_LEVELS	fcst.field.level
FCST_VAR<n>_THRESH	fcst.field.cat_thresh
FCST_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the forecast field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_FCST_CONV_RADIUS}

METplus Config(s)	MET Config File
MTD_FCST_CONV_RADIUS	fcst.conv_radius

${METPLUS_FCST_CONV_THRESH}

METplus Config(s)	MET Config File
MTD_FCST_CONV_THRESH	fcst.conv_thresh

${METPLUS_OBS_FILE_TYPE}

METplus Config(s)	MET Config File
OBS_MTD_INPUT_DATATYPE	obs.file_type

${METPLUS_OBS_FIELD}

METplus Config(s)	MET Config File
OBS_VAR<n>_NAME	fcst.field.name
OBS_VAR<n>_LEVELS	fcst.field.level
OBS_VAR<n>_THRESH	fcst.field.cat_thresh
OBS_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the observation field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_OBS_CONV_RADIUS}

METplus Config(s)	MET Config File
MTD_OBS_CONV_RADIUS	obs.conv_radius

${METPLUS_OBS_CONV_THRESH}

METplus Config(s)	MET Config File
MTD_OBS_CONV_THRESH	obs.conv_thresh

${METPLUS_MIN_VOLUME}

METplus Config(s)	MET Config File
MTD_MIN_VOLUME	min_volume

${METPLUS_OUTPUT_PREFIX}

METplus Config(s)	MET Config File
MTD_OUTPUT_PREFIX	output_prefix

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
MTD_MET_CONFIG_OVERRIDES	n/a

6.16 PB2NC

6.16.1 Description

The PB2NC wrapper is a Python script that encapsulates the behavior of the MET pb2nc tool to convert prepBUFR files into netCDF.

6.16.2 METplus Configuration

PB2NC_INPUT_DIR
PB2NC_OUTPUT_DIR
PB2NC_INPUT_TEMPLATE
PB2NC_OUTPUT_TEMPLATE
PB2NC_SKIP_IF_OUTPUT_EXISTS
PB2NC_OFFSETS
PB2NC_INPUT_DATATYPE
PB2NC_CONFIG_FILE
PB2NC_MESSAGE_TYPE
PB2NC_STATION_ID
PB2NC_GRID
PB2NC_POLY
PB2NC_OBS_BUFR_VAR_LIST
PB2NC_TIME_SUMMARY_FLAG
PB2NC_TIME_SUMMARY_BEG
PB2NC_TIME_SUMMARY_END
PB2NC_TIME_SUMMARY_VAR_NAMES
PB2NC_TIME_SUMMARY_TYPES
PB2NC_OBS_WINDOW_BEGIN
PB2NC_OBS_WINDOW_END
PB2NC_VALID_BEGIN
PB2NC_VALID_END
PB2NC_CUSTOM_LOOP_LIST
PB2NC_MET_CONFIG_OVERRIDES
PB2NC_PB_REPORT_TYPE
PB2NC_LEVEL_RANGE_BEG
PB2NC_LEVEL_RANGE_END
PB2NC_LEVEL_CATEGORY
PB2NC_QUALITY_MARK_THRESH
PB2NC_OBS_BUFR_MAP

Warning: DEPRECATED:

PREPBUFR_DATA_DIR
PREPBUFR_MODEL_DIR_NAME
PREPBUFR_DIR_REGEX
PREPBUFR_FILE_REGEX
NC_FILE_TMPL
PB2NC_VERTICAL_LEVEL
OBS_BUFR_VAR_LIST
TIME_SUMMARY_FLAG
TIME_SUMMARY_BEG
TIME_SUMMARY_END
TIME_SUMMARY_VAR_NAMES
TIME_SUMMARY_TYPES
OVERWRITE_NC_OUTPUT
VERTICAL_LOCATION

6.16.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/PB2NCConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// PB2NC configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// PrepBufr message type
//
${METPLUS_MESSAGE_TYPE}
```

(continues on next page)

(continued from previous page)

```
//
// Mapping of message type group name to comma-separated list of values
// Derive PRMSL only for SURFACE message types
//
message_type_group_map = [
   { key = "SURFACE"; val = "ADPSFC,SFCSHP,MSONET";               },
   { key = "ANYAIR";  val = "AIRCAR,AIRCFT";                      },
   { key = "ANYSFC";  val = "ADPSFC,SFCSHP,ADPUPA,PROFLR,MSONET"; },
   { key = "ONLYSF";  val = "ADPSFC,SFCSHP";                      }
];

//
// Mapping of input PrepBufr message types to output message types
//
message_type_map = [];

//
// PrepBufr station ID
//
${METPLUS_STATION_ID}


////////////////////////////////////////////////////////////////////////////////

//
// Observation time window
//
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observation retention regions
//
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observing location elevation
//
elevation_range = {
   beg =  -1000;
   end = 100000;
```

(continues on next page)

(continued from previous page)

```
}

////////////////////////////////////////////////////////////////////////////////

//
// Observation types
//
//pb_report_type =
${METPLUS_PB_REPORT_TYPE}

in_report_type  = [];

instrument_type = [];

////////////////////////////////////////////////////////////////////////////////

//
// Vertical levels to retain
//
//level_range = {
${METPLUS_LEVEL_RANGE_DICT}

//level_category =
${METPLUS_LEVEL_CATEGORY}

////////////////////////////////////////////////////////////////////////////////

//
// BUFR variable names to retain or derive.
// If emtpy, process all available variables.
//
${METPLUS_OBS_BUFR_VAR}

////////////////////////////////////////////////////////////////////////////////

//
// Mapping of BUFR variable name to GRIB name. The default map is defined at
// obs_prepbufr_map. This replaces/expends the default map.
//
//obs_bufr_map =
${METPLUS_OBS_BUFR_MAP}

// This map is for PREPBUFR. It will be added into obs_bufr_map.
// Please do not override this map.
//obs_prepbufr_map =
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//quality_mark_thresh =
${METPLUS_QUALITY_MARK_THRESH}

event_stack_flag    = TOP;

////////////////////////////////////////////////////////////////////////////////
//
// Time periods for the summarization
//
${METPLUS_TIME_SUMMARY_DICT}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

//version = "V9.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MESSAGE_TYPE}

METplus Config(s)	MET Config File
PB2NC_MESSAGE_TYPE	message_type

${METPLUS_STATION_ID}

METplus Config(s)	MET Config File
PB2NC_STATION_ID	station_id

${METPLUS_OBS_WINDOW_DICT}

METplus Config(s)	MET Config File
PB2NC_OBS_WINDOW_BEGIN	obs_window.beg
PB2NC_OBS_WINDOW_END	obs_window.end

${METPLUS_MASK_DICT}

METplus Config(s)	MET Config File
PB2NC_MASK_GRID	mask.grid
PB2NC_MASK_POLY	mask.poly

Note: Since the default value in the MET config file for 'grid' is grid = ["FULL"];, setting GRID_STAT_MASK_GRID to an empty string will result in a value of grid = []; in the MET config file.

${METPLUS_OBS_BUFR_VAR}

METplus Config(s)	MET Config File
PB2NC_OBS_BUFR_VAR_LIST	obs_bufr_var

${METPLUS_TIME_SUMMARY_DICT}

METplus Config(s)	MET Config File
PB2NC_TIME_SUMMARY_FLAG	time_summary.flag
PB2NC_TIME_SUMMARY_RAW_DATA	time_summary.raw_data
PB2NC_TIME_SUMMARY_BEG	time_summary.beg
PB2NC_TIME_SUMMARY_END	time_summary.end
PB2NC_TIME_SUMMARY_STEP	time_summary.step
PB2NC_TIME_SUMMARY_WIDTH	time_summary.width
PB2NC_TIME_SUMMARY_GRIB_CODES	time_summary.grib_code
PB2NC_TIME_SUMMARY_VAR_NAMES	time_summary.obs_var
PB2NC_TIME_SUMMARY_TYPES	time_summary.type
PB2NC_TIME_SUMMARY_VALID_FREQ	time_summary.vld_freq
PB2NC_TIME_SUMMARY_VALID_THRESH	time_summary.vld_thresh

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
PB2NC_MET_CONFIG_OVERRIDES	n/a

${METPLUS_PB_REPORT_TYPE}

METplus Config(s)	MET Config File
PB2NC_PB_REPORT_TYPE	pb_report_type

${METPLUS_LEVEL_RANGE_DICT}

METplus Config(s)	MET Config File
PB2NC_LEVEL_RANGE_BEG	level_range.beg
PB2NC_LEVEL_RANGE_END	level_range.end

${METPLUS_LEVEL_CATEGORY}

METplus Config(s)	MET Config File
PB2NC_LEVEL_CATEGORY	level_category

${METPLUS_QUALITY_MARK_THRESH}

METplus Config(s)	MET Config File
PB2NC_QUALITY_MARK_THRESH	quality_mark_thresh

${METPLUS_OBS_BUFR_MAP}

METplus Config(s)	MET Config File
PB2NC_OBS_BUFR_MAP	obs_bufr_map

6.17 PCPCombine

6.17.1 Description

The PCPCombine wrapper is a Python script that encapsulates the MET PCPCombine tool. It provides the infrastructure to combine or extract from files to build desired accumulations.

PCPCombine wrapper can be configured to process forecast and/or observation data. Setting *FCST_PCP_COMBINE_RUN* = True will process forecast data and setting *OBS_PCP_COMBINE_RUN* = True will process observation data.

PCPCombine wrapper can be configured to build a command for the sum, add, subtract, and derive methods using *FCST_PCP_COMBINE_METHOD* and/or *OBS_PCP_COMBINE_METHOD*. Each method executes logic to gather the desired input files to build the command based on specific examples.

6.17.1.1 Accumulations

The desired accumulation to build is defined using *FCST_PCP_COMBINE_OUTPUT_ACCUM* or *OBS_PCP_COMBINE_OUTPUT_ACCUM*. The default units are hours unless otherwise specified. The output field name can be set explicitly using *FCST_PCP_COMBINE_OUTPUT_NAME* or *OBS_PCP_COMBINE_OUTPUT_NAME*.

For the ADD and DERIVE methods, the input accumulation(s) can be specified using *FCST_PCP_COMBINE_INPUT_ACCUMS* or *OBS_PCP_COMBINE_INPUT_ACCUMS*. The default units are hours unless otherwise specified. This can be a list of accumulation amounts in order of preference. If the remaining accumulation needed to build the desired accumulation is less than the first accumulation, then the next value in the list will be used. The name and level of the field to read for each input accumulation can be specified with *FCST_PCP_COMBINE_INPUT_NAMES*/*FCST_PCP_COMBINE_INPUT_LEVELS* or *OBS_PCP_COMBINE_INPUT_NAMES*/*OBS_PCP_COMBINE_INPUT_LEVELS*. These lists must be the same length as *FCST_PCP_COMBINE_INPUT_ACCUMS* or *OBS_PCP_COMBINE_INPUT_ACCUMS*.

6.17.1.2 Constant Initialization Time

For the ADD and DERIVE methods, *FCST_PCP_COMBINE_CONSTANT_INIT* or *OBS_PCP_COMBINE_CONSTANT_INIT* can be set to **True** to gather input files that all contain the same initialization time.

6.17.1.3 User-Defined Commands

There are many ways to utilize PCPCombine that may not align with the logic used to gather files. If this is the case, then the method can be set to **USER_DEFINED** and the explicit command arguments can be specified using *FCST_PCP_COMBINE_COMMAND* or *OBS_PCP_COMBINE_COMMAND*. Other METplus configuration variables and filename template tags can be referenced in the explicit command. Note that the path to the pcp_combine executable and the output path should not be included in the command value. The output path is controlled by *FCST_PCP_COMBINE_INPUT_TEMPLATE*/*FCST_PCP_COMBINE_INPUT_DIR* or *OBS_PCP_COMBINE_INPUT_TEMPLATE*/*OBS_PCP_COMBINE_INPUT_DIR* and will automatically be added to the end of the command.

6.17.2 METplus Configuration

FCST_PCP_COMBINE_INPUT_DIR
FCST_PCP_COMBINE_OUTPUT_DIR
OBS_PCP_COMBINE_INPUT_DIR
OBS_PCP_COMBINE_OUTPUT_DIR
FCST_PCP_COMBINE_INPUT_TEMPLATE
FCST_PCP_COMBINE_OUTPUT_TEMPLATE
OBS_PCP_COMBINE_INPUT_TEMPLATE
OBS_PCP_COMBINE_OUTPUT_TEMPLATE
LOG_PCP_COMBINE_VERBOSITY
FCST_PCP_COMBINE_INPUT_ACCUMS
FCST_PCP_COMBINE_INPUT_NAMES
FCST_PCP_COMBINE_INPUT_LEVELS
FCST_PCP_COMBINE_INPUT_OPTIONS
OBS_PCP_COMBINE_INPUT_ACCUMS
OBS_PCP_COMBINE_INPUT_NAMES
OBS_PCP_COMBINE_INPUT_LEVELS
OBS_PCP_COMBINE_INPUT_OPTIONS
FCST_PCP_COMBINE_INPUT_DATATYPE
OBS_PCP_COMBINE_INPUT_DATATYPE
FCST_PCP_COMBINE_RUN
OBS_PCP_COMBINE_RUN
FCST_PCP_COMBINE_METHOD
OBS_PCP_COMBINE_METHOD
FCST_PCP_COMBINE_MIN_FORECAST

OBS_PCP_COMBINE_MIN_FORECAST
FCST_PCP_COMBINE_MAX_FORECAST
OBS_PCP_COMBINE_MAX_FORECAST
FCST_PCP_COMBINE_BUCKET_INTERVAL
OBS_PCP_COMBINE_BUCKET_INTERVAL
FCST_PCP_COMBINE_CONSTANT_INIT
OBS_PCP_COMBINE_CONSTANT_INIT
FCST_PCP_COMBINE_STAT_LIST
OBS_PCP_COMBINE_STAT_LIST
PCP_COMBINE_SKIP_IF_OUTPUT_EXISTS
FCST_PCP_COMBINE_COMMAND
OBS_PCP_COMBINE_COMMAND
PCP_COMBINE_CUSTOM_LOOP_LIST
FCST_PCP_COMBINE_LOOKBACK
OBS_PCP_COMBINE_LOOKBACK
FCST_PCP_COMBINE_USE_ZERO_ACCUM
OBS_PCP_COMBINE_USE_ZERO_ACCUM
FCST_PCP_COMBINE_EXTRA_NAMES
FCST_PCP_COMBINE_EXTRA_LEVELS
FCST_PCP_COMBINE_EXTRA_OUTPUT_NAMES
OBS_PCP_COMBINE_EXTRA_NAMES
OBS_PCP_COMBINE_EXTRA_LEVELS
OBS_PCP_COMBINE_EXTRA_OUTPUT_NAMES
FCST_PCP_COMBINE_OUTPUT_ACCUM
FCST_PCP_COMBINE_OUTPUT_NAME
OBS_PCP_COMBINE_OUTPUT_ACCUM
OBS_PCP_COMBINE_OUTPUT_NAME

Warning: DEPRECATED:

PCP_COMBINE_METHOD
FCST_MIN_FORECAST
FCST_MAX_FORECAST
OBS_MIN_FORECAST
OBS_MAX_FORECAST
FCST_DATA_INTERVAL
OBS_DATA_INTERVAL
FCST_IS_DAILY_FILE

OBS_IS_DAILY_FILE
FCST_TIMES_PER_FILE
OBS_TIMES_PER_FILE
FCST_LEVEL
OBS_LEVEL
FCST_PCP_COMBINE_INPUT_LEVEL
OBS_PCP_COMBINE_INPUT_LEVEL
FCST_PCP_COMBINE_<n>_FIELD_NAME
OBS_PCP_COMBINE_<n>_FIELD_NAME
FCST_PCP_COMBINE_DATA_INTERVAL
OBS_PCP_COMBINE_DATA_INTERVAL
FCST_PCP_COMBINE_TIMES_PER_FILE
OBS_PCP_COMBINE_TIMES_PER_FILE
FCST_PCP_COMBINE_IS_DAILY_FILE
OBS_PCP_COMBINE_IS_DAILY_FILE
FCST_PCP_COMBINE_DERIVE_LOOKBACK
OBS_PCP_COMBINE_DERIVE_LOOKBACK

6.18 PlotDataPlane

6.18.1 Description

The PlotDataPlane wrapper is a Python script that encapsulates the MET plot_data_plane tool. It provides the infrastructure to read in any input that MET can read and plot them. This tool is often used to verify that the data is mapped to the correct grid location.

6.18.2 Configuration

PLOT_DATA_PLANE_INPUT_DIR
PLOT_DATA_PLANE_OUTPUT_DIR
PLOT_DATA_PLANE_INPUT_TEMPLATE
PLOT_DATA_PLANE_OUTPUT_TEMPLATE
PLOT_DATA_PLANE_FIELD_NAME
PLOT_DATA_PLANE_FIELD_LEVEL
PLOT_DATA_PLANE_FIELD_EXTRA
LOG_PLOT_DATA_PLANE_VERBOSITY
PLOT_DATA_PLANE_TITLE
PLOT_DATA_PLANE_COLOR_TABLE
PLOT_DATA_PLANE_RANGE_MIN_MAX

PLOT_DATA_PLANE_CONVERT_TO_IMAGE
PLOT_DATA_PLANE_SKIP_IF_OUTPUT_EXISTS

6.19 PlotPointObs

6.19.1 Description

The PlotPointObs wrapper is a Python script that encapsulates the MET plot_point_obs tool. It provides the infrastructure to read in any input that MET can read and plot them.

6.19.2 Configuration

PLOT_POINT_OBS_RUNTIME_FREQ
PLOT_POINT_OBS_INPUT_TEMPLATE
PLOT_POINT_OBS_INPUT_DIR
PLOT_POINT_OBS_GRID_INPUT_TEMPLATE
PLOT_POINT_OBS_GRID_INPUT_DIR
PLOT_POINT_OBS_OUTPUT_TEMPLATE
PLOT_POINT_OBS_OUTPUT_DIR
PLOT_POINT_OBS_SKIP_IF_OUTPUT_EXISTS
PLOT_POINT_OBS_TITLE
LOG_PLOT_POINT_OBS_VERBOSITY
PLOT_POINT_OBS_GRID_DATA_FIELD
PLOT_POINT_OBS_GRID_DATA_REGRID_TO_GRID
PLOT_POINT_OBS_GRID_DATA_REGRID_METHOD
PLOT_POINT_OBS_GRID_DATA_REGRID_WIDTH
PLOT_POINT_OBS_GRID_DATA_REGRID_VLD_THRESH
PLOT_POINT_OBS_GRID_DATA_REGRID_SHAPE
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_COLOR_TABLE
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_PLOT_MIN
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_PLOT_MAX
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_COLORBAR_FLAG
PLOT_POINT_OBS_MSG_TYP
PLOT_POINT_OBS_SID_INC
PLOT_POINT_OBS_SID_EXC
PLOT_POINT_OBS_OBS_VAR
PLOT_POINT_OBS_OBS_GC
PLOT_POINT_OBS_OBS_QUALITY
PLOT_POINT_OBS_VALID_BEG
PLOT_POINT_OBS_VALID_END
PLOT_POINT_OBS_LAT_THRESH

PLOT_POINT_OBS_LON_THRESH
PLOT_POINT_OBS_ELV_THRESH
PLOT_POINT_OBS_HGT_THRESH
PLOT_POINT_OBS_PRS_THRESH
PLOT_POINT_OBS_OBS_THRESH
PLOT_POINT_OBS_CENSOR_THRESH
PLOT_POINT_OBS_CENSOR_VAL
PLOT_POINT_OBS_DOTSIZE
PLOT_POINT_OBS_LINE_COLOR
PLOT_POINT_OBS_LINE_WIDTH
PLOT_POINT_OBS_FILL_COLOR
PLOT_POINT_OBS_FILL_PLOT_INFO_FLAG
PLOT_POINT_OBS_FILL_PLOT_INFO_COLOR_TABLE
PLOT_POINT_OBS_FILL_PLOT_INFO_PLOT_MIN
PLOT_POINT_OBS_FILL_PLOT_INFO_PLOT_MAX
PLOT_POINT_OBS_FILL_PLOT_INFO_COLORBAR_FLAG
PLOT_POINT_OBS_POINT_DATA
PLOT_POINT_OBS_MET_CONFIG_OVERRIDES

6.19.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/PlotPointObsConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// Plot-Point-Obs configuration file.
//
// For additional information, please see the MET Users Guide.
//
////////////////////////////////////////////////////////////////////////////////

// Gridded data plotting options

//grid_data = {
${METPLUS_GRID_DATA_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

// Point data filtering options
// May be set separately in each "point_data" entry

//msg_typ =
${METPLUS_MSG_TYP}

//sid_inc =
${METPLUS_SID_INC}

//sid_exc =
${METPLUS_SID_EXC}

//obs_var =
${METPLUS_OBS_VAR}

//obs_gc =
${METPLUS_OBS_GC}

//obs_quality =
${METPLUS_OBS_QUALITY}

//valid_beg =
${METPLUS_VALID_BEG}

//valid_end =
${METPLUS_VALID_END}

//lat_thresh =
${METPLUS_LAT_THRESH}

//lon_thresh =
${METPLUS_LON_THRESH}

//elv_thresh =
${METPLUS_ELV_THRESH}

//hgt_thresh =
${METPLUS_HGT_THRESH}

//prs_thresh =
${METPLUS_PRS_THRESH}
```

(continues on next page)

(continued from previous page)

```
//obs_thresh =
${METPLUS_OBS_THRESH}

// Point data pre-processing options
// May be set separately in each "point_data" entry

//convert(x)    = x;

//censor_thresh =
${METPLUS_CENSOR_THRESH}

//censor_val =
${METPLUS_CENSOR_VAL}


// Point data plotting options
// May be set separately in each "point_data" entry

//dotsize =
${METPLUS_DOTSIZE}

//line_color =
${METPLUS_LINE_COLOR}

//line_width =
${METPLUS_LINE_WIDTH}

//fill_color =
${METPLUS_FILL_COLOR}

//fill_plot_info = {
${METPLUS_FILL_PLOT_INFO_DICT}

// Array of point data filtering, pre-processing, and plotting options
//point_data =
${METPLUS_POINT_DATA}

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_GRID_DATA_DICT}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_GRID_DATA_FIELD	grid_data.field
PLOT_POINT_OBS_GRID_DATA_REGRID_TO_GRID	grid_data.regrid.to_grid
PLOT_POINT_OBS_GRID_DATA_REGRID_METHOD	grid_data.regrid.method
PLOT_POINT_OBS_GRID_DATA_REGRID_WIDTH	grid_data.regrid.width
PLOT_POINT_OBS_GRID_DATA_REGRID_VLD_THRESH	grid_data.regrid.vld_thresh
PLOT_POINT_OBS_GRID_DATA_REGRID_SHAPE	grid_data.regrid.shape
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_COLOR_TABLE	grid_data.grid_plot_info.color_table
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_PLOT_MIN	grid_data.grid_plot_info.plot_min
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_PLOT_MAX	grid_data.grid_plot_info.plot_max
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_COLORBAR_FLAG	grid_data.grid_plot_info.colorbar_flag

${METPLUS_MSG_TYP}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_MSG_TYP	msg_typ

${METPLUS_SID_INC}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_SID_INC	sid_inc

${METPLUS_SID_EXC}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_SID_EXC	sid_exc

${METPLUS_OBS_VAR}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_OBS_VAR	obs_var

${METPLUS_OBS_GC}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_OBS_GC	obs_gc

${METPLUS_OBS_QUALITY}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_OBS_QUALITY	obs_quality

${METPLUS_VALID_BEG}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_VALID_BEG	valid_beg

${METPLUS_VALID_END}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_VALID_END	valid_end

${METPLUS_LAT_THRESH}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_LAT_THRESH	lat_thresh

${METPLUS_LON_THRESH}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_LON_THRESH	lon_thresh

${METPLUS_ELV_THRESH}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_ELV_THRESH	elv_thresh

${METPLUS_HGT_THRESH}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_HGT_THRESH	hgt_thresh

${METPLUS_PRS_THRESH}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_PRS_THRESH	prs_thresh

${METPLUS_OBS_THRESH}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_OBS_THRESH	obs_thresh

${METPLUS_CENSOR_THRESH}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_CENSOR_THRESH	censor_thresh

${METPLUS_CENSOR_VAL}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_CENSOR_VAL	censor_val

${METPLUS_DOTSIZE}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_DOTSIZE	dotsize

${METPLUS_LINE_COLOR}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_LINE_COLOR	line_color

${METPLUS_LINE_WIDTH}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_LINE_WIDTH	line_width

${METPLUS_FILL_COLOR}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_FILL_COLOR	fill_color

${METPLUS_FILL_PLOT_INFO_DICT}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_FILL_PLOT_INFO_FLAG	fill_plot_info.flag
PLOT_POINT_OBS_FILL_PLOT_INFO_COLOR_TABLE	fill_plot_info.color_table
PLOT_POINT_OBS_FILL_PLOT_INFO_PLOT_MIN	fill_plot_info.plot_min
PLOT_POINT_OBS_FILL_PLOT_INFO_PLOT_MAX	fill_plot_info.plot_max
PLOT_POINT_OBS_FILL_PLOT_INFO_COLORBAR_FLAG	fill_plot_info.colorbar_flag

${METPLUS_POINT_DATA}

METplus Config(s)	MET Config File
PLOT_POINT_OBS_POINT_DATA	point_data

6.20 Point2Grid

6.20.1 Description

The Point2Grid wrapper is a Python script that encapsulates the MET point2grid tool. It provides the infrastructure to read in point observations and place them on a grid

6.20.2 METplus Configuration

POINT2GRID_INPUT_DIR
POINT2GRID_OUTPUT_DIR
POINT2GRID_INPUT_TEMPLATE
POINT2GRID_OUTPUT_TEMPLATE
POINT2GRID_WINDOW_BEGIN
POINT2GRID_WINDOW_END
POINT2GRID_REGRID_TO_GRID
POINT2GRID_INPUT_FIELD
POINT2GRID_INPUT_LEVEL
POINT2GRID_QC_FLAGS
POINT2GRID_ADP
POINT2GRID_REGRID_METHOD
POINT2GRID_GAUSSIAN_DX
POINT2GRID_GAUSSIAN_RADIUS
POINT2GRID_PROB_CAT_THRESH
POINT2GRID_VLD_THRESH
POINT2GRID_CUSTOM_LOOP_LIST
POINT2GRID_SKIP_IF_OUTPUT_EXISTS

6.21 PointStat

6.21.1 Description

The PointStat wrapper is a Python script that encapsulates the MET point_stat tool. It provides the infrastructure to read in gridded model data and netCDF point observation data to perform grid-to-point (grid-to-obs) verification.

6.21.2 Configuration

FCST_POINT_STAT_INPUT_DIR
OBS_POINT_STAT_INPUT_DIR
POINT_STAT_OUTPUT_DIR
FCST_POINT_STAT_INPUT_TEMPLATE
OBS_POINT_STAT_INPUT_TEMPLATE
POINT_STAT_VERIFICATION_MASK_TEMPLATE
POINT_STAT_OUTPUT_PREFIX
LOG_POINT_STAT_VERBOSITY
POINT_STAT_OFFSETS
FCST_POINT_STAT_INPUT_DATATYPE
OBS_POINT_STAT_INPUT_DATATYPE
POINT_STAT_FCST_FILE_TYPE
POINT_STAT_OBS_FILE_TYPE
POINT_STAT_CONFIG_FILE
MODEL
POINT_STAT_REGRID_TO_GRID
POINT_STAT_REGRID_METHOD
POINT_STAT_REGRID_WIDTH
POINT_STAT_REGRID_VLD_THRESH
POINT_STAT_REGRID_SHAPE
POINT_STAT_REGRID_CONVERT
POINT_STAT_REGRID_CENSOR_THRESH
POINT_STAT_REGRID_CENSOR_VAL
POINT_STAT_MASK_GRID
POINT_STAT_MASK_POLY
POINT_STAT_MASK_SID
POINT_STAT_MASK_LLPNT
POINT_STAT_MESSAGE_TYPE
POINT_STAT_CUSTOM_LOOP_LIST
POINT_STAT_SKIP_IF_OUTPUT_EXISTS
POINT_STAT_DESC
POINT_STAT_MET_CONFIG_OVERRIDES
POINT_STAT_CLIMO_CDF_BINS
POINT_STAT_CLIMO_CDF_CENTER_BINS
POINT_STAT_CLIMO_CDF_WRITE_BINS
POINT_STAT_CLIMO_CDF_DIRECT_PROB
POINT_STAT_OBS_QUALITY_INC
POINT_STAT_OBS_QUALITY_EXC
POINT_STAT_OUTPUT_FLAG_FHO
POINT_STAT_OUTPUT_FLAG_CTC

POINT_STAT_OUTPUT_FLAG_CTS
POINT_STAT_OUTPUT_FLAG_MCTC
POINT_STAT_OUTPUT_FLAG_MCTS
POINT_STAT_OUTPUT_FLAG_CNT
POINT_STAT_OUTPUT_FLAG_SL1L2
POINT_STAT_OUTPUT_FLAG_SAL1L2
POINT_STAT_OUTPUT_FLAG_VL1L2
POINT_STAT_OUTPUT_FLAG_VAL1L2
POINT_STAT_OUTPUT_FLAG_VCNT
POINT_STAT_OUTPUT_FLAG_PCT
POINT_STAT_OUTPUT_FLAG_PSTD
POINT_STAT_OUTPUT_FLAG_PJC
POINT_STAT_OUTPUT_FLAG_PRC
POINT_STAT_OUTPUT_FLAG_ECNT
POINT_STAT_OUTPUT_FLAG_ORANK
POINT_STAT_OUTPUT_FLAG_RPS
POINT_STAT_OUTPUT_FLAG_ECLV
POINT_STAT_OUTPUT_FLAG_MPR
POINT_STAT_OUTPUT_FLAG_SEEPS
POINT_STAT_OUTPUT_FLAG_SEEPS_MPR
POINT_STAT_INTERP_VLD_THRESH
POINT_STAT_INTERP_SHAPE
POINT_STAT_INTERP_TYPE_METHOD
POINT_STAT_INTERP_TYPE_WIDTH
POINT_STAT_CLIMO_MEAN_FILE_NAME
POINT_STAT_CLIMO_MEAN_VAR<n>_NAME
POINT_STAT_CLIMO_MEAN_VAR<n>_LEVELS
POINT_STAT_CLIMO_MEAN_VAR<n>_OPTIONS
POINT_STAT_CLIMO_MEAN_FIELD
POINT_STAT_CLIMO_MEAN_REGRID_METHOD
POINT_STAT_CLIMO_MEAN_REGRID_WIDTH
POINT_STAT_CLIMO_MEAN_REGRID_VLD_THRESH
POINT_STAT_CLIMO_MEAN_REGRID_SHAPE
POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD
POINT_STAT_CLIMO_MEAN_MATCH_MONTH
POINT_STAT_CLIMO_MEAN_DAY_INTERVAL
POINT_STAT_CLIMO_MEAN_HOUR_INTERVAL
POINT_STAT_CLIMO_MEAN_USE_FCST
POINT_STAT_CLIMO_MEAN_USE_OBS
POINT_STAT_CLIMO_STDEV_FILE_NAME
POINT_STAT_CLIMO_STDEV_VAR<n>_NAME

POINT_STAT_CLIMO_STDEV_VAR<n>_LEVELS
POINT_STAT_CLIMO_STDEV_VAR<n>_OPTIONS
POINT_STAT_CLIMO_STDEV_FIELD
POINT_STAT_CLIMO_STDEV_REGRID_METHOD
POINT_STAT_CLIMO_STDEV_REGRID_WIDTH
POINT_STAT_CLIMO_STDEV_REGRID_VLD_THRESH
POINT_STAT_CLIMO_STDEV_REGRID_SHAPE
POINT_STAT_CLIMO_STDEV_TIME_INTERP_METHOD
POINT_STAT_CLIMO_STDEV_MATCH_MONTH
POINT_STAT_CLIMO_STDEV_DAY_INTERVAL
POINT_STAT_CLIMO_STDEV_HOUR_INTERVAL
POINT_STAT_CLIMO_STDEV_USE_FCST
POINT_STAT_CLIMO_STDEV_USE_OBS
POINT_STAT_HSS_EC_VALUE
POINT_STAT_HIRA_FLAG
POINT_STAT_HIRA_WIDTH
POINT_STAT_HIRA_VLD_THRESH
POINT_STAT_HIRA_COV_THRESH
POINT_STAT_HIRA_SHAPE
POINT_STAT_HIRA_PROB_CAT_THRESH
POINT_STAT_MESSAGE_TYPE_GROUP_MAP
FCST_POINT_STAT_IS_PROB
FCST_POINT_STAT_PROB_IN_GRIB_PDS
FCST_POINT_STAT_WINDOW_BEGIN
FCST_POINT_STAT_WINDOW_END
OBS_POINT_STAT_WINDOW_BEGIN
OBS_POINT_STAT_WINDOW_END
POINT_STAT_NEIGHBORHOOD_WIDTH
POINT_STAT_NEIGHBORHOOD_SHAPE
FCST_POINT_STAT_VAR<n>_NAME
FCST_POINT_STAT_VAR<n>_LEVELS
FCST_POINT_STAT_VAR<n>_THRESH
FCST_POINT_STAT_VAR<n>_OPTIONS
OBS_POINT_STAT_VAR<n>_NAME
OBS_POINT_STAT_VAR<n>_LEVELS
OBS_POINT_STAT_VAR<n>_THRESH
OBS_POINT_STAT_VAR<n>_OPTIONS
POINT_STAT_OBS_VALID_BEG
POINT_STAT_OBS_VALID_END
POINT_STAT_SEEPS_P1_THRESH

Warning: DEPRECATED:

FCST_INPUT_DIR
OBS_INPUT_DIR
START_HOUR
END_HOUR
BEG_TIME
FCST_HR_START
FCST_HR_END
FCST_HR_INTERVAL
OBS_INPUT_DIR_REGEX
FCST_INPUT_DIR_REGEX
FCST_INPUT_FILE_REGEX
OBS_INPUT_FILE_REGEX
OBS_INPUT_FILE_TMPL
FCST_INPUT_FILE_TMPL
REGRID_TO_GRID
CLIMO_POINT_STAT_INPUT_DIR
CLIMO_POINT_STAT_INPUT_TEMPLATE
POINT_STAT_CLIMO_MEAN_INPUT_DIR
POINT_STAT_CLIMO_STDEV_INPUT_DIR
POINT_STAT_CLIMO_MEAN_INPUT_TEMPLATE
POINT_STAT_CLIMO_STDEV_INPUT_TEMPLATE

6.21.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/PointStatConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
```

(continues on next page)

(continued from previous page)

```
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

//
// Forecast and observation fields to be verified
//
```

(continues on next page)

(continued from previous page)

```
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
//interp = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_DESC}

METplus Config(s)	MET Config File
DESC -or- *POINT_STAT_DESC*	desc

${METPLUS_REGRID_DICT}

METplus Config(s)	MET Config File
POINT_STAT_REGRID_SHAPE	regrid.shape
POINT_STAT_REGRID_METHOD	regrid.method
POINT_STAT_REGRID_WIDTH	regrid.width
POINT_STAT_REGRID_VLD_THRESH	regrid.vld_thresh
POINT_STAT_REGRID_TO_GRID	regrid.to_grid
POINT_STAT_REGRID_CONVERT	regrid.convert
POINT_STAT_REGRID_CENSOR_THRESH	regrid.censor_thresh
POINT_STAT_REGRID_CENSOR_VAL	regrid.censor_val

${METPLUS_FCST_FIELD}

METplus Config(s)	MET Config File
FCST_VAR<n>_NAME	fcst.field.name
FCST_VAR<n>_LEVELS	fcst.field.level
FCST_VAR<n>_THRESH	fcst.field.cat_thresh
FCST_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the forecast field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_FCST_FILE_TYPE}

METplus Config(s)	MET Config File
POINT_STAT_FCST_FILE_TYPE	fcst.file_type

${METPLUS_OBS_FIELD}

METplus Config(s)	MET Config File
OBS_VAR<n>_NAME	obs.field.name
OBS_VAR<n>_LEVELS	obs.field.level
OBS_VAR<n>_THRESH	obs.field.cat_thresh
OBS_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the observation field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_OBS_FILE_TYPE}

METplus Config(s)	MET Config File
POINT_STAT_OBS_FILE_TYPE	obs.file_type

${METPLUS_MESSAGE_TYPE}

METplus Config(s)	MET Config File
POINT_STAT_MESSAGE_TYPE	message_type

${METPLUS_CLIMO_MEAN_DICT}

METplus Config(s)	MET Config File
POINT_STAT_CLIMO_MEAN_FILE_NAME	climo_mean.file_name
POINT_STAT_CLIMO_MEAN_FIELD	climo_mean.field
POINT_STAT_CLIMO_MEAN_REGRID_METHOD	climo_mean.regrid.method
POINT_STAT_CLIMO_MEAN_REGRID_WIDTH	climo_mean.regrid.width
POINT_STAT_CLIMO_MEAN_REGRID_VLD_THRESH	climo_mean.regrid.vld_thresh
POINT_STAT_CLIMO_MEAN_REGRID_SHAPE	climo_mean.regrid.shape
POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD	climo_mean.time_interp_method
POINT_STAT_CLIMO_MEAN_MATCH_MONTH	climo_mean.match_month
POINT_STAT_CLIMO_MEAN_DAY_INTERVAL	climo_mean.day_interval
POINT_STAT_CLIMO_MEAN_HOUR_INTERVAL	climo_mean.hour_interval

${METPLUS_CLIMO_STDEV_DICT}

METplus Config(s)	MET Config File
POINT_STAT_CLIMO_STDEV_FILE_NAME	climo_stdev.file_name
POINT_STAT_CLIMO_STDEV_FIELD	climo_stdev.field
POINT_STAT_CLIMO_STDEV_REGRID_METHOD	climo_stdev.regrid.method
POINT_STAT_CLIMO_STDEV_REGRID_WIDTH	climo_stdev.regrid.width
POINT_STAT_CLIMO_STDEV_REGRID_VLD_THRESH	climo_stdev.regrid.vld_thresh
POINT_STAT_CLIMO_STDEV_REGRID_SHAPE	climo_stdev.regrid.shape
POINT_STAT_CLIMO_STDEV_TIME_INTERP_METHOD	climo_stdev.time_interp_method
POINT_STAT_CLIMO_STDEV_MATCH_MONTH	climo_stdev.match_month
POINT_STAT_CLIMO_STDEV_DAY_INTERVAL	climo_stdev.day_interval
POINT_STAT_CLIMO_STDEV_HOUR_INTERVAL	climo_stdev.hour_interval

${METPLUS_OBS_WINDOW_DICT}

METplus Config(s)	MET Config File
OBS_WINDOW_BEGIN	obs_window.beg
OBS_WINDOW_END	obs_window.end

${METPLUS_MASK_DICT}

METplus Config(s)	MET Config File
POINT_STAT_MASK_GRID	mask.grid
POINT_STAT_MASK_POLY	mask.poly
POINT_STAT_MASK_SID	mask.sid
POINT_STAT_MASK_LLPNT	mask.llpnt

${METPLUS_OUTPUT_PREFIX}

METplus Config(s)	MET Config File
POINT_STAT_OUTPUT_PREFIX	output_prefix

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
POINT_STAT_MET_CONFIG_OVERRIDES	n/a

${METPLUS_CLIMO_CDF_DICT}

METplus Config(s)	MET Config File
POINT_STAT_CLIMO_CDF_BINS	climo_cdf.cdf_bins
POINT_STAT_CLIMO_CDF_CENTER_BINS	climo_cdf.center_bins
POINT_STAT_CLIMO_CDF_WRITE_BINS	climo_cdf.write_bins
POINT_STAT_CLIMO_CDF_DIRECT_PROB	climo_cdf.direct_prob

${METPLUS_OBS_QUALITY_INC}

METplus Config(s)	MET Config File
POINT_STAT_OBS_QUALITY_INC	obs_quality_inc

${METPLUS_OBS_QUALITY_EXC}

METplus Config(s)	MET Config File
POINT_STAT_OBS_QUALITY_EXC	obs_quality_exc

${METPLUS_OUTPUT_FLAG_DICT}

METplus Config(s)	MET Config File
POINT_STAT_OUTPUT_FLAG_FHO	output_flag.fho
POINT_STAT_OUTPUT_FLAG_CTC	output_flag.ctc
POINT_STAT_OUTPUT_FLAG_CTS	output_flag.cts
POINT_STAT_OUTPUT_FLAG_MCTC	output_flag.mctc
POINT_STAT_OUTPUT_FLAG_MCTS	output_flag.mcts
POINT_STAT_OUTPUT_FLAG_CNT	output_flag.cnt
POINT_STAT_OUTPUT_FLAG_SL1L2	output_flag.sl1l2
POINT_STAT_OUTPUT_FLAG_SAL1L2	output_flag.sal1l2
POINT_STAT_OUTPUT_FLAG_VL1L2	output_flag.vl1l2
POINT_STAT_OUTPUT_FLAG_VAL1L2	output_flag.val1l2
POINT_STAT_OUTPUT_FLAG_VCNT	output_flag.vcnt
POINT_STAT_OUTPUT_FLAG_PCT	output_flag.pct
POINT_STAT_OUTPUT_FLAG_PSTD	output_flag.pstd
POINT_STAT_OUTPUT_FLAG_PJC	output_flag.pjc
POINT_STAT_OUTPUT_FLAG_PRC	output_flag.prc
POINT_STAT_OUTPUT_FLAG_ECNT	output_flag.ecnt
POINT_STAT_OUTPUT_FLAG_ORANK	output_flag.orank
POINT_STAT_OUTPUT_FLAG_RPS	output_flag.rps
POINT_STAT_OUTPUT_FLAG_ECLV	output_flag.eclv
POINT_STAT_OUTPUT_FLAG_MPR	output_flag.mpr
POINT_STAT_OUTPUT_FLAG_SEEPS	output_flag.seeps
POINT_STAT_OUTPUT_FLAG_SEEPS_MPR	output_flag.seeps_mpr

${METPLUS_INTERP_DICT}

METplus Config(s)	MET Config File
POINT_STAT_INTERP_VLD_THRESH	interp.vld_thresh
POINT_STAT_INTERP_SHAPE	interp.shape
POINT_STAT_INTERP_TYPE_METHOD	interp.type.method
POINT_STAT_INTERP_TYPE_WIDTH	interp.type.width

${METPLUS_HSS_EC_VALUE}

METplus Config(s)	MET Config File
POINT_STAT_HSS_EC_VALUE	hss_ec_value

${METPLUS_HIRA_DICT}

METplus Config(s)	MET Config File
POINT_STAT_HIRA_FLAG	hira.flag
POINT_STAT_HIRA_WIDTH	hira.width
POINT_STAT_HIRA_VLD_THRESH	hira.vld_thresh
POINT_STAT_HIRA_COV_THRESH	hira.cov_thresh
POINT_STAT_HIRA_SHAPE	hira.shape
POINT_STAT_HIRA_PROB_CAT_THRESH	hira.prob_cat_thresh

${METPLUS_MESSAGE_TYPE_GROUP_MAP}

METplus Config(s)	MET Config File
POINT_STAT_MESSAGE_TYPE_GROUP_MAP	message_type_group_map

${METPLUS_SEEPS_P1_THRESH}

METplus Config(s)	MET Config File
POINT_STAT_SEEPS_P1_THRESH	seeps_p1_thresh

6.22 PyEmbedIngest

6.22.1 Description

Used to configure the PyEmbedIngest wrapper that runs RegridDataPlane to convert data using python embedding scripts into NetCDF so it can be read by the MET tools.

6.22.2 METplus Configuration

PY_EMBED_INGEST_<n>_OUTPUT_DIR
PY_EMBED_INGEST_<n>_OUTPUT_TEMPLATE
PY_EMBED_INGEST_<n>_SCRIPT
PY_EMBED_INGEST_<n>_TYPE
PY_EMBED_INGEST_<n>_OUTPUT_GRID
PY_EMBED_INGEST_CUSTOM_LOOP_LIST
PY_EMBED_INGEST_<n>_OUTPUT_FIELD_NAME
PY_EMBED_INGEST_SKIP_IF_OUTPUT_EXISTS

Warning: DEPRECATED:

CUSTOM_INGEST_<n>_OUTPUT_DIR
CUSTOM_INGEST_<n>_OUTPUT_TEMPLATE
CUSTOM_INGEST_<n>_SCRIPT
CUSTOM_INGEST_<n>_TYPE
CUSTOM_INGEST_<n>_OUTPUT_GRID

6.23 RegridDataPlane

6.23.1 Description

Used to configure the MET tool regrid_data_plane which can be used to change projections of a grid with user configurable interpolation choices. It can also be used to convert GRIB1 and GRIB2 files into netcdf files if desired.

6.23.2 METplus Configuration

FCST_REGRID_DATA_PLANE_INPUT_DIR
OBS_REGRID_DATA_PLANE_INPUT_DIR
FCST_REGRID_DATA_PLANE_INPUT_TEMPLATE
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE
FCST_REGRID_DATA_PLANE_OUTPUT_TEMPLATE
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE
FCST_REGRID_DATA_PLANE_TEMPLATE
OBS_REGRID_DATA_PLANE_TEMPLATE
FCST_REGRID_DATA_PLANE_RUN
OBS_REGRID_DATA_PLANE_RUN
REGRID_DATA_PLANE_SKIP_IF_OUTPUT_EXISTS
REGRID_DATA_PLANE_VERIF_GRID
FCST_REGRID_DATA_PLANE_INPUT_DATATYPE
OBS_REGRID_DATA_PLANE_INPUT_DATATYPE
REGRID_DATA_PLANE_GAUSSIAN_DX
REGRID_DATA_PLANE_GAUSSIAN_RADIUS
REGRID_DATA_PLANE_WIDTH
REGRID_DATA_PLANE_METHOD
REGRID_DATA_PLANE_CUSTOM_LOOP_LIST
REGRID_DATA_PLANE_ONCE_PER_FIELD
FCST_REGRID_DATA_PLANE_VAR<n>_INPUT_FIELD_NAME
FCST_REGRID_DATA_PLANE_VAR<n>_INPUT_LEVEL
FCST_REGRID_DATA_PLANE_VAR<n>_OUTPUT_FIELD_NAME

OBS_REGRID_DATA_PLANE_VAR<n>_INPUT_FIELD_NAME
OBS_REGRID_DATA_PLANE_VAR<n>_INPUT_LEVEL
OBS_REGRID_DATA_PLANE_VAR<n>_OUTPUT_FIELD_NAME

Warning: DEPRECATED:

VERIFICATION_GRID

6.24 SeriesAnalysis

6.24.1 Description

The SeriesAnalysis wrapper is used to find files and build a command that calls the MET tool SeriesAnalysis. It can be configured to process ranges of inputs, i.e. once for all files, once for each forecast lead (using , once for a group of forecast leads, once for each initialization time, etc. with the *SERIES_ANALYSIS_RUNTIME_FREQ* variable. Optionally, a .tcst file generated by TCStat can be provided to allow filtering by storm ID (see *SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID*). Images of the output data can also optionally be generated as well as animated gif images (See *SERIES_ANALYSIS_GENERATE_PLOTS* and *SERIES_ANALYSIS_GENERATE_ANIMATIONS*)

6.24.2 METplus Configuration

LOG_SERIES_ANALYSIS_VERBOSITY
SERIES_ANALYSIS_CONFIG_FILE
SERIES_ANALYSIS_RUNTIME_FREQ
SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID
SERIES_ANALYSIS_BACKGROUND_MAP
SERIES_ANALYSIS_REGRID_TO_GRID
SERIES_ANALYSIS_REGRID_METHOD
SERIES_ANALYSIS_REGRID_WIDTH
SERIES_ANALYSIS_REGRID_VLD_THRESH
SERIES_ANALYSIS_REGRID_SHAPE
SERIES_ANALYSIS_REGRID_CONVERT
SERIES_ANALYSIS_REGRID_CENSOR_THRESH
SERIES_ANALYSIS_REGRID_CENSOR_VAL
SERIES_ANALYSIS_STAT_LIST
SERIES_ANALYSIS_IS_PAIRED

SERIES_ANALYSIS_CUSTOM_LOOP_LIST
SERIES_ANALYSIS_SKIP_IF_OUTPUT_EXISTS
SERIES_ANALYSIS_GENERATE_PLOTS (Optional)
SERIES_ANALYSIS_GENERATE_ANIMATIONS (Optional)
PLOT_DATA_PLANE_TITLE (Optional)
LEAD_SEQ_<n> (Optional)
LEAD_SEQ_<n>_LABEL (Optional)
SERIES_ANALYSIS_DESC
SERIES_ANALYSIS_CAT_THRESH
SERIES_ANALYSIS_VLD_THRESH
SERIES_ANALYSIS_BLOCK_SIZE
SERIES_ANALYSIS_CTS_LIST
FCST_SERIES_ANALYSIS_PROB_THRESH
SERIES_ANALYSIS_MET_CONFIG_OVERRIDES
FCST_SERIES_ANALYSIS_INPUT_DIR
OBS_SERIES_ANALYSIS_INPUT_DIR
BOTH_SERIES_ANALYSIS_INPUT_DIR
SERIES_ANALYSIS_TC_STAT_INPUT_DIR
SERIES_ANALYSIS_OUTPUT_DIR
FCST_SERIES_ANALYSIS_INPUT_TEMPLATE
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE
BOTH_SERIES_ANALYSIS_INPUT_TEMPLATE
FCST_SERIES_ANALYSIS_INPUT_FILE_LIST
OBS_SERIES_ANALYSIS_INPUT_FILE_LIST
BOTH_SERIES_ANALYSIS_INPUT_FILE_LIST
SERIES_ANALYSIS_TC_STAT_INPUT_TEMPLATE
SERIES_ANALYSIS_OUTPUT_TEMPLATE
SERIES_ANALYSIS_CLIMO_MEAN_FILE_NAME
SERIES_ANALYSIS_CLIMO_MEAN_VAR<n>_NAME
SERIES_ANALYSIS_CLIMO_MEAN_VAR<n>_LEVELS
SERIES_ANALYSIS_CLIMO_MEAN_VAR<n>_OPTIONS
SERIES_ANALYSIS_CLIMO_MEAN_FIELD
SERIES_ANALYSIS_CLIMO_MEAN_REGRID_METHOD
SERIES_ANALYSIS_CLIMO_MEAN_REGRID_WIDTH
SERIES_ANALYSIS_CLIMO_MEAN_REGRID_VLD_THRESH
SERIES_ANALYSIS_CLIMO_MEAN_REGRID_SHAPE
SERIES_ANALYSIS_CLIMO_MEAN_TIME_INTERP_METHOD
SERIES_ANALYSIS_CLIMO_MEAN_MATCH_MONTH
SERIES_ANALYSIS_CLIMO_MEAN_DAY_INTERVAL
SERIES_ANALYSIS_CLIMO_MEAN_HOUR_INTERVAL
SERIES_ANALYSIS_CLIMO_MEAN_FILE_TYPE

Warning: DEPRECATED:

SERIES_ANALYSIS_CLIMO_MEAN_INPUT_DIR
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_TEMPLATE
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_TEMPLATE

6.24.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/SeriesAnalysisConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// Series-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
//
//desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
//obtype =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

censor_thresh = [];
censor_val    = [];
//cat_thresh =
${METPLUS_CAT_THRESH}
cnt_thresh    = [ NA ];
cnt_logic     = UNION;

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_CAT_THRESH}
   ${METPLUS_FCST_FIELD}
}
obs = {
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}
```

(continues on next page)

(continued from previous page)

```
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

//
// Number of grid points to be processed concurrently.  Set smaller to use
// less memory but increase the number of passes through the data.
//
//block_size =
${METPLUS_BLOCK_SIZE}

//
// Ratio of valid matched pairs to compute statistics for a grid point
//
//vld_thresh =
${METPLUS_VLD_THRESH}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
```

(continues on next page)

(continued from previous page)

```
//output_stats = {
${METPLUS_OUTPUT_STATS_DICT}

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_DESC}

METplus Config(s)	MET Config File
DESC -or- *SERIES_ANALYSIS_DESC*	desc

${METPLUS_OBTYPE}

METplus Config(s)	MET Config File
OBTYPE	obtype

${METPLUS_REGRID_DICT}

METplus Config(s)	MET Config File
SERIES_ANALYSIS_REGRID_SHAPE	regrid.shape
SERIES_ANALYSIS_REGRID_METHOD	regrid.method
SERIES_ANALYSIS_REGRID_WIDTH	regrid.width
SERIES_ANALYSIS_REGRID_VLD_THRESH	regrid.vld_thresh
SERIES_ANALYSIS_REGRID_TO_GRID	regrid.to_grid
SERIES_ANALYSIS_REGRID_CONVERT	regrid.convert
SERIES_ANALYSIS_REGRID_CENSOR_THRESH	regrid.censor_thresh
SERIES_ANALYSIS_REGRID_CENSOR_VAL	regrid.censor_val

${METPLUS_CAT_THRESH}

METplus Config(s)	MET Config File
SERIES_ANALYSIS_CAT_THRESH	cat_thresh

${METPLUS_FCST_FILE_TYPE}

METplus Config(s)	MET Config File
FCST_SERIES_ANALYSIS_INPUT_DATATYPE	fcst.file_type

${METPLUS_FCST_FIELD}

METplus Config(s)	MET Config File
FCST_VAR<n>_NAME	fcst.field.name
FCST_VAR<n>_LEVELS	fcst.field.level
FCST_VAR<n>_THRESH	fcst.field.cat_thresh
FCST_VAR<n>_OPTIONS	n/a
FCST_SERIES_ANALYSIS_PROB_THRESH	n/a

Note: For more information on controlling the forecast field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_OBS_FILE_TYPE}

METplus Config(s)	MET Config File
OBS_SERIES_ANALYSIS_INPUT_DATATYPE	obs.file_type

${METPLUS_OBS_FIELD}

METplus Config(s)	MET Config File
OBS_VAR<n>_NAME	fcst.field.name
OBS_VAR<n>_LEVELS	fcst.field.level
OBS_VAR<n>_THRESH	fcst.field.cat_thresh
OBS_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the observation field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_CLIMO_MEAN_DICT}

METplus Config(s)	MET Config File
SERIES_ANALYSIS_CLIMO_MEAN_FILE_NAME	climo_mean.file_name
SERIES_ANALYSIS_CLIMO_MEAN_FIELD	climo_mean.field
SERIES_ANALYSIS_CLIMO_MEAN_REGRID_METHOD	climo_mean.regrid.method
SERIES_ANALYSIS_CLIMO_MEAN_REGRID_WIDTH	climo_mean.regrid.width
SERIES_ANALYSIS_CLIMO_MEAN_REGRID_VLD_THRESH	climo_mean.regrid.vld_thresh
SERIES_ANALYSIS_CLIMO_MEAN_REGRID_SHAPE	climo_mean.regrid.shape
SERIES_ANALYSIS_CLIMO_MEAN_TIME_INTERP_METHOD	climo_mean.time_interp_method
SERIES_ANALYSIS_CLIMO_MEAN_MATCH_MONTH	climo_mean.match_month
SERIES_ANALYSIS_CLIMO_MEAN_DAY_INTERVAL	climo_mean.day_interval
SERIES_ANALYSIS_CLIMO_MEAN_HOUR_INTERVAL	climo_mean.hour_interval
SERIES_ANALYSIS_CLIMO_MEAN_FILE_TYPE	climo_mean.file_type

${METPLUS_CLIMO_STDEV_DICT}

METplus Config(s)	MET Config File
SERIES_ANALYSIS_CLIMO_STDEV_FILE_NAME	climo_stdev.file_name
SERIES_ANALYSIS_CLIMO_STDEV_FIELD	climo_stdev.field
SERIES_ANALYSIS_CLIMO_STDEV_REGRID_METHOD	climo_stdev.regrid.method
SERIES_ANALYSIS_CLIMO_STDEV_REGRID_WIDTH	climo_stdev.regrid.width
SERIES_ANALYSIS_CLIMO_STDEV_REGRID_VLD_THRESH	climo_stdev.regrid.vld_thresh
SERIES_ANALYSIS_CLIMO_STDEV_REGRID_SHAPE	climo_stdev.regrid.shape
SERIES_ANALYSIS_CLIMO_STDEV_TIME_INTERP_METHOD	climo_stdev.time_interp_method
SERIES_ANALYSIS_CLIMO_STDEV_MATCH_MONTH	climo_stdev.match_month
SERIES_ANALYSIS_CLIMO_STDEV_DAY_INTERVAL	climo_stdev.day_interval
SERIES_ANALYSIS_CLIMO_STDEV_HOUR_INTERVAL	climo_stdev.hour_interval
SERIES_ANALYSIS_CLIMO_STDEV_FILE_TYPE	climo_stdev.file_type

${METPLUS_CLIMO_CDF_DICT}

METplus Config(s)	MET Config File
SERIES_ANALYSIS_CLIMO_CDF_BINS	climo_cdf.cdf_bins
SERIES_ANALYSIS_CLIMO_CDF_CENTER_BINS	climo_cdf.center_bins
SERIES_ANALYSIS_CLIMO_CDF_DIRECT_PROB	climo_cdf.direct_prob

${METPLUS_MASK_DICT}

METplus Config(s)	MET Config File
SERIES_ANALYSIS_MASK_GRID	mask.grid
SERIES_ANALYSIS_MASK_POLY	mask.poly

${METPLUS_BLOCK_SIZE}

METplus Config(s)	MET Config File
SERIES_ANALYSIS_BLOCK_SIZE	block_size

${METPLUS_VLD_THRESH}

METplus Config(s)	MET Config File
SERIES_ANALYSIS_VLD_THRESH	vld_thresh

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
SERIES_ANALYSIS_MET_CONFIG_OVERRIDES	n/a

${METPLUS_HSS_EC_VALUE}

METplus Config(s)	MET Config File
SERIES_ANALYSIS_HSS_EC_VALUE	hss_ec_value

${METPLUS_OUTPUT_STATS_DICT}

METplus Config(s)	MET Config File
SERIES_ANALYSIS_OUTPUT_STATS_FHO	output_stats.fho
SERIES_ANALYSIS_OUTPUT_STATS_CTC	output_stats.ctc
SERIES_ANALYSIS_OUTPUT_STATS_CTS	output_stats.cts
SERIES_ANALYSIS_OUTPUT_STATS_MCTC	output_stats.mctc
SERIES_ANALYSIS_OUTPUT_STATS_MCTS	output_stats.mcts
SERIES_ANALYSIS_OUTPUT_STATS_CNT	output_stats.cnt
SERIES_ANALYSIS_OUTPUT_STATS_SL1L2	output_stats.sl1l2
SERIES_ANALYSIS_OUTPUT_STATS_SAL1L2	output_stats.sal1l2
SERIES_ANALYSIS_OUTPUT_STATS_PCT	output_stats.pct
SERIES_ANALYSIS_OUTPUT_STATS_PSTD	output_stats.pstd
SERIES_ANALYSIS_OUTPUT_STATS_PJC	output_stats.pjc
SERIES_ANALYSIS_OUTPUT_STATS_PRC	output_stats.prc

${METPLUS_FCST_CAT_THRESH}

METplus Config(s)	MET Config File
FCST_SERIES_ANALYSIS_CAT_THRESH	fcst.cat_thresh

${METPLUS_OBS_CAT_THRESH}

METplus Config(s)	MET Config File
OBS_SERIES_ANALYSIS_CAT_THRESH	obs.cat_thresh

6.25 SeriesByInit

6.25.1 Description

Warning: This tool has been DEPRECATED. Please use SeriesAnalysis wrapper

6.26 SeriesByLead

6.26.1 Description

Warning: This tool has been DEPRECATED. Please use SeriesAnalysis wrapper

6.27 StatAnalysis

6.27.1 Description

The StatAnalysis wrapper encapsulates the behavior of the MET stat_analysis tool. It provides the infrastructure to summarize and filter the MET .stat files.

6.27.1.1 Timing

This wrapper is configured differently than many of the other wrappers that loop over multiple run times. The StatAnalysis wrapper is designed to process a range of run times at once using filtering to subset what is processed. The VALID_BEG and VALID_END or INIT_BEG and INIT_END variables are used to calculate filtering criteria.

Prior to v5.0.0, only the year, month, and day (YYYYMMDD) of the init/valid begin and end times were read by the wrapper. The hours, minutes, and seconds were ignored to be filtered using FCST_HOUR_LIST and OBS_HOUR_LIST. Now the full time information is read and to enable users to process a more specific range of time. To preserve the original behavior, end times that do not include hours, minutes, or seconds will process up to 23:59:59 on that day unless specific hours are defined with FCST_HOUR_LIST or OBS_HOUR_LIST.

Note: The LEAD_SEQ variable that typically defines a list of forecast leads to process is not used by the wrapper. Instead the FCST_LEAD_LIST and OBS_LEAD_LIST are used to filter out forecast leads from the data.

6.27.1.2 Optional MET Configuration File

The wrapped MET config file specified with *STAT_ANALYSIS_CONFIG_FILE* is optional in the StatAnalysis wrapper. Excluding this option will result in a call to stat_analysis with the job arguments added via the command line. Only 1 job can be defined in no wrapped MET configuration file is used. To use a configuration file, set the following in the METplus config file:

```
STAT_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/STATAnalysisConfig_wrapped
```

6.27.1.3 Jobs

The job arguments can be defined by setting *STAT_ANALYSIS_JOB<n>* variables, e.g. STAT_ANALYSIS_JOB1. All of the job commands including the -job argument are set here. Prior to v5.0.0, the config variables STAT_ANALYSIS_JOB_NAME and STAT_ANALYSIS_JOB_ARGS were used to set the value following the -job argument and any other job arguments respectively.

Multiple jobs can be defined as of v5.0.0 using STAT_ANALYSIS_JOB1, STAT_ANALYSIS_JOB2, etc. All jobs will be passed to each call to stat_analysis. Only 1 job can be specified if no MET config file is set with *STAT_ANALYSIS_CONFIG_FILE*.

6.27.1.4 Filtering with Lists

There are many configuration variables that end with _LIST that control settings in the STATAnalysisConfig_wrapped file. For example, MODEL_LIST controls the model variable in the MET config file and FCST_LEAD_LIST controls the fcst_lead variable. The value for each of these _LIST variables can be a list of values separated by comma. The value of GROUP_LIST_ITEMS is a comma-separated list of _LIST variable names that will be grouped together for each call to stat_analysis. The value of LOOP_LIST_ITEMS is a comma-separated list of _LIST variable names that will be looped over to create multiple calls to stat_analysis. The tool will be called with every combination of the LOOP_LIST_ITEMS list values. List variables that are not included in either GROUP_LIST_ITEMS or LOOP_LIST_ITEMS will be automatically added to GROUP_LIST_ITEMS. Lists defined in LOOP_LIST_ITEMS that are empty lists will be automatically moved to GROUP_LIST_ITEMS.

6.27.1.5 Looping Over Groups of Lists

New in v5.0.0 is the ability to define groups of list items that can be looped over. For example, a user may want to process forecast leads 1-3 in a single run, then process forecast leads 4-6 in the next. To accomplish this, define each group of items in a separate config variable ending with a number. Then add the name of the list (without the numbers) to LOOP_LIST_ITEMS:

```
[config]
FCST_LEAD_LIST1 = 1,2,3
FCST_LEAD_LIST2 = 4,5,6
LOOP_LIST_ITEMS = FCST_LEAD_LIST
```

If FCST_LEAD_LIST was added to GROUP_LIST_ITEMS instead, then all 6 items defined in the 2 lists will be combined and passed to the tool at once.

6.27.1.6 Filtering Begin and End Times

Starting in v5.0.0, the [fcst/obs]_[init/valid]_[beg/end] in the wrapped MET config file can be set using the corresponding METplus config variables. The values can include the filename template tags that are supported in the wrapper (see *Additional Filename Template Tags* (page 241)). For example, to set the fcst_valid_beg value:

```
[config]
VALID_BEG = 20221014
STAT_ANALYSIS_FCST_VALID_BEG = {fcst_valid_beg?fmt=%Y%m%d_%H%M%S}
```

This will set fcst_valid_beg = "20221014_000000"; in the MET config file.

Prior to v5.0.0, settings hour values in [FCST/OBS]_[INIT/VALID]_HOUR_LIST would result in the corresponding _beg and _end values in the wrapped MET config file to be set based on the hours and the [INIT/VALID]_[BEG/END] values.

6.27.1.7 Additional Filename Template Tags

The StatAnalysis wrapper supports additional tags that can be substituted into the input and output paths because the wrapper processes a range of time.

The following filename template tags can be used:

- model
- desc
- vx_mask
- interp_mthd
- interp_pnts
- cov_thresh
- alpha
- line_type
- fcst_var
- obs_var
- fcst_units
- obs_units
- fcst_thresh
- obs_thresh

- fcst_level
- obs_level
- fcst_valid_hour
- obs_valid_hour
- fcst_init_hour
- obs_init_hour
- fcst_lead
- obs_lead
- fcst_valid_hour_beg
- fcst_valid_hour_end
- obs_valid_hour_beg
- obs_valid_hour_end
- fcst_init_hour_beg
- fcst_init_hour_end
- obs_init_hour_beg
- obs_init_hour_end
- valid_hour
- valid_hour_beg
- valid_hour_end
- init_hour
- init_hour_beg
- init_hour_end
- fcst_valid
- fcst_valid_beg
- fcst_valid_end
- fcst_init
- fcst_init_beg
- fcst_init_end
- obs_valid
- obs_valid_beg
- obs_valid_end
- obs_init

- obs_init_beg
- obs_init_end
- valid
- valid_beg
- valid_end
- init
- init_beg
- init_end
- fcst_lead
- fcst_lead_hour
- fcst_lead_min
- fcst_lead_sec
- fcst_lead_totalsec
- obs_lead
- obs_lead_hour
- obs_lead_min
- obs_lead_sec
- obs_lead_totalsec
- lead
- lead_hour
- lead_min
- lead_sec
- lead_totalsec

Please note that some of these items will be set to an empty string depending on the configuration. For example, lead_hour, lead_min, lead_sec, and lead_totalsec cannot be computed if there are multiple leads being processed in a given run. Another example, if fcst_valid_beg has the same value as fcst_valid_end, then fcst_valid will be set to the same value, otherwise it will be left as an empty string.

6.27.1.8 Outputs

This wrapper can be configured to write 3 types of output files. Output files specified with the -out command line argument can be defined by setting *STAT_ANALYSIS_OUTPUT_TEMPLATE* and optionally *STAT_ANALYSIS_OUTPUT_DIR*. Output files specified with the -dump_row or -out_stat arguments must be defined in a job using *STAT_ANALYSIS_JOB<n>*. The [dump_row_file] keyword can be added to a job after the -dump_row argument only if a *MODEL<n>_STAT_ANALYSIS_DUMP_ROW_TEMPLATE* is set. Similarly, the [out_stat_file] keyword can be added to a job after the -out_stat argument only if a *MODEL<n>_STAT_ANALYSIS_OUT_STAT_TEMPLATE* is set.

6.27.2 METplus Configuration

The following values **must** be defined in the METplus configuration file:

STAT_ANALYSIS_JOB<n>
STAT_ANALYSIS_OUTPUT_DIR
MODEL<n>
MODEL<n>_STAT_ANALYSIS_LOOKIN_DIR
GROUP_LIST_ITEMS
LOOP_LIST_ITEMS

The following values are optional in the METplus configuration file:

STAT_ANALYSIS_CONFIG_FILE
LOG_STAT_ANALYSIS_VERBOSITY
STAT_ANALYSIS_CUSTOM_LOOP_LIST
MODEL<n>_OBTYPE
VAR<n>_FOURIER_DECOMP
VAR<n>_WAVE_NUM_LIST
MODEL_LIST
DESC_LIST
FCST_LEAD_LIST
OBS_LEAD_LIST
FCST_VALID_HOUR_LIST
FCST_INIT_HOUR_LIST
OBS_VALID_HOUR_LIST
OBS_INIT_HOUR_LIST
FCST_VAR_LIST
OBS_VAR_LIST
FCST_UNITS_LIST
OBS_UNITS_LIST

FCST_LEVEL_LIST
OBS_LEVEL_LIST
VX_MASK_LIST
INTERP_MTHD_LIST
INTERP_PNTS_LIST
FCST_THRESH_LIST
OBS_THRESH_LIST
COV_THRESH_LIST
ALPHA_LIST
LINE_TYPE_LIST
STAT_ANALYSIS_HSS_EC_VALUE
STAT_ANALYSIS_OUTPUT_TEMPLATE
MODEL<n>_STAT_ANALYSIS_DUMP_ROW_TEMPLATE
MODEL<n>_STAT_ANALYSIS_OUT_STAT_TEMPLATE
STAT_ANALYSIS_FCST_INIT_BEG
STAT_ANALYSIS_FCST_INIT_END
STAT_ANALYSIS_FCST_VALID_BEG
STAT_ANALYSIS_FCST_VALID_END
STAT_ANALYSIS_OBS_INIT_BEG
STAT_ANALYSIS_OBS_INIT_END
STAT_ANALYSIS_OBS_VALID_BEG
STAT_ANALYSIS_OBS_VALID_END
STAT_ANALYSIS_MET_CONFIG_OVERRIDES

Warning: DEPRECATED:

STAT_ANALYSIS_LOOKIN_DIR
STAT_ANALYSIS_OUT_DIR
STAT_ANALYSIS_CONFIG
VALID_HOUR_METHOD
VALID_HOUR_BEG
VALID_HOUR_END
VALID_HOUR_INCREMENT
INIT_HOUR_BEG
INIT_HOUR_END
INIT_HOUR_INCREMENT
MODEL
OBTYPE
JOB_NAME

JOB_ARGS
FCST_LEAD
FCST_VAR_NAME
FCST_VAR_LEVEL
OBS_VAR_NAME
OBS_VAR_LEVEL
REGION
INTERP
INTERP_PTS
FCST_THRESH
COV_THRESH
LINE_TYPE
STAT_ANALYSIS_DUMP_ROW_TMPL
STAT_ANALYSIS_OUT_STAT_TMPL
PLOT_TIME
MODEL<n>_NAME
MODEL<n>_OBS_NAME
MODEL<n>_NAME_ON_PLOT
MODEL<n>_STAT_DIR
REGION_LIST
LEAD_LIST
STAT_ANALYSIS_JOB_NAME
STAT_ANALYSIS_JOB_ARGS

6.27.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/STATAnalysisConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// STAT-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Filtering input STAT lines by the contents of each column
//
${METPLUS_MODEL}
${METPLUS_DESC}

${METPLUS_FCST_LEAD}
${METPLUS_OBS_LEAD}

${METPLUS_FCST_VALID_BEG}
${METPLUS_FCST_VALID_END}
${METPLUS_FCST_VALID_HOUR}

${METPLUS_OBS_VALID_BEG}
${METPLUS_OBS_VALID_END}
${METPLUS_OBS_VALID_HOUR}

${METPLUS_FCST_INIT_BEG}
${METPLUS_FCST_INIT_END}
${METPLUS_FCST_INIT_HOUR}

${METPLUS_OBS_INIT_BEG}
${METPLUS_OBS_INIT_END}
${METPLUS_OBS_INIT_HOUR}

${METPLUS_FCST_VAR}
${METPLUS_OBS_VAR}

${METPLUS_FCST_UNITS}
${METPLUS_OBS_UNITS}

${METPLUS_FCST_LEVEL}
${METPLUS_OBS_LEVEL}

${METPLUS_OBTYPE}

${METPLUS_VX_MASK}

${METPLUS_INTERP_MTHD}

${METPLUS_INTERP_PNTS}

${METPLUS_FCST_THRESH}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OBS_THRESH}
${METPLUS_COV_THRESH}

${METPLUS_ALPHA}

${METPLUS_LINE_TYPE}

column = [];

weight = [];

////////////////////////////////////////////////////////////////////////////////

//
// Array of STAT-Analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
out_alpha = 0.05;

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// WMO mean computation logic
//
wmo_sqrt_stats   = [ "CNT:FSTDEV",  "CNT:OSTDEV",  "CNT:ESTDEV",
                     "CNT:RMSE",    "CNT:RMSFA",   "CNT:RMSOA",
                     "VCNT:FS_RMS", "VCNT:OS_RMS", "VCNT:RMSVE",
                     "VCNT:FSTDEV", "VCNT:OSTDEV" ];

wmo_fisher_stats = [ "CNT:PR_CORR", "CNT:SP_CORR",
                     "CNT:KT_CORR", "CNT:ANOM_CORR" ];
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;
vif_flag       = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_DESC}

METplus Config(s)	MET Config File
DESC_LIST	desc

${METPLUS_FCST_LEAD}

METplus Config(s)	MET Config File
FCST_LEAD_LIST	fcst_lead

${METPLUS_OBS_LEAD}

METplus Config(s)	MET Config File
OBS_LEAD_LIST	obs_lead

${METPLUS_FCST_VALID_BEG}

METplus Config(s)	MET Config File
FCST_VALID_HOUR_LIST and *VALID_BEG*	fcst_valid_beg

${METPLUS_FCST_VALID_END}

METplus Config(s)	MET Config File
FCST_VALID_HOUR_LIST and *VALID_END*	fcst_valid_end

${METPLUS_FCST_VALID_HOUR}

METplus Config(s)	MET Config File
FCST_VALID_HOUR_LIST	fcst_valid_hour

${METPLUS_OBS_VALID_BEG}

METplus Config(s)	MET Config File
OBS_VALID_HOUR_LIST and *VALID_BEG*	obs_valid_beg

${METPLUS_OBS_VALID_END}

METplus Config(s)	MET Config File
OBS_VALID_HOUR_LIST and *VALID_END*	obs_valid_end

${METPLUS_OBS_VALID_HOUR}

METplus Config(s)	MET Config File
OBS_VALID_HOUR_LIST	obs_valid_hour

${METPLUS_FCST_INIT_BEG}

METplus Config(s)	MET Config File
FCST_INIT_HOUR_LIST and *INIT_BEG*	fcst_init_beg

${METPLUS_FCST_INIT_END}

METplus Config(s)	MET Config File
FCST_INIT_HOUR_LIST and *INIT_END*	fcst_init_end

${METPLUS_FCST_INIT_HOUR}

METplus Config(s)	MET Config File
FCST_INIT_HOUR_LIST	fcst_init_hour

${METPLUS_OBS_INIT_BEG}

METplus Config(s)	MET Config File
OBS_INIT_HOUR_LIST and *INIT_BEG*	obs_init_beg

${METPLUS_OBS_INIT_END}

METplus Config(s)	MET Config File
OBS_INIT_HOUR_LIST and *INIT_END*	obs_init_end

${METPLUS_OBS_INIT_HOUR}

METplus Config(s)	MET Config File
OBS_INIT_HOUR_LIST	obs_init_hour

${METPLUS_FCST_VAR}

METplus Config(s)	MET Config File
FCST_VAR_LIST	fcst_var

${METPLUS_OBS_VAR}

METplus Config(s)	MET Config File
OBS_VAR_LIST	obs_var

${METPLUS_FCST_UNITS}

METplus Config(s)	MET Config File
FCST_UNITS_LIST	fcst_units

${METPLUS_OBS_UNITS}

METplus Config(s)	MET Config File
OBS_UNITS_LIST	obs_units

${METPLUS_FCST_LEVEL}

METplus Config(s)	MET Config File
FCST_LEVEL_LIST	fcst_lev

${METPLUS_OBS_LEVEL}

METplus Config(s)	MET Config File
OBS_LEVEL_LIST	obs_lev

${METPLUS_OBTYPE}

METplus Config(s)	MET Config File
MODEL<n>_OBTYPE	obtype

${METPLUS_VX_MASK}

METplus Config(s)	MET Config File
VX_MASK_LIST	vx_mask

${METPLUS_INTERP_MTHD}

METplus Config(s)	MET Config File
INTERP_MTHD_LIST	interp_mthd

${METPLUS_INTERP_PNTS}

METplus Config(s)	MET Config File
INTERP_PNTS_LIST	interp_pnts

${METPLUS_FCST_THRESH}

METplus Config(s)	MET Config File
FCST_THRESH_LIST	fcst_thresh

${METPLUS_OBS_THRESH}

METplus Config(s)	MET Config File
OBS_THRESH_LIST	obs_thresh

${METPLUS_COV_THRESH}

METplus Config(s)	MET Config File
COV_THRESH_LIST	cov_thresh

${METPLUS_ALPHA}

METplus Config(s)	MET Config File
ALPHA_LIST	alpha

${METPLUS_LINE_TYPE}

METplus Config(s)	MET Config File
LINE_TYPE_LIST	line_type

${METPLUS_JOBS}

METplus Config(s)	MET Config File
STAT_ANALYSIS_JOB_NAME	jobs

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
STAT_ANALYSIS_MET_CONFIG_OVERRIDES	n/a

${METPLUS_HSS_EC_VALUE}

METplus Config(s)	MET Config File
STAT_ANALYSIS_HSS_EC_VALUE	hss_ec_value

6.28 TCDiag

6.28.1 Description

The TC-Diag wrapper encapsulates the behavior of the MET tc_diag tool. It provides the infrastructure to compute diagnostics from model fields and tracks. It can be configured to run over a single intialization time, all of the initialization times for a given storm, or over many storms. Configuration also allows a user to select which domain(s) of the input model data to use in the diagnostics calculations, set which levels and variables will be used as well as details about the azimuth-range grid used for the calculations, and to control which output files are generated. Future functionality of the tc_diag tool, such as vortex removal, will also be configurable from this wrapper.

6.28.2 METplus Configuration

TC_DIAG_INPUT_DIR
TC_DIAG_DECK_INPUT_DIR
TC_DIAG_OUTPUT_DIR
TC_DIAG_DECK_TEMPLATE
TC_DIAG_INPUT_TEMPLATE
TC_DIAG_INPUT_FILE_LIST
TC_DIAG_OUTPUT_TEMPLATE
LOG_TC_DIAG_VERBOSITY
TC_DIAG_CONFIG_FILE
TC_DIAG_MODEL
TC_DIAG_STORM_ID
TC_DIAG_BASIN
TC_DIAG_CYCLONE
TC_DIAG_INIT_INCLUDE
TC_DIAG_VALID_BEG
TC_DIAG_VALID_END
TC_DIAG_VALID_INCLUDE
TC_DIAG_VALID_EXCLUDE
TC_DIAG_VALID_HOUR
TC_DIAG_DIAG_SCRIPT
TC_DIAG_DOMAIN_INFO<n>_DOMAIN
TC_DIAG_DOMAIN_INFO<n>_N_RANGE
TC_DIAG_DOMAIN_INFO<n>_N_AZIMUTH

TC_DIAG_DOMAIN_INFO<n>_DELTA_RANGE_KM
TC_DIAG_DOMAIN_INFO<n>_DIAG_SCRIPT
TC_DIAG_CENSOR_THRESH
TC_DIAG_CENSOR_VAL
TC_DIAG_CONVERT
TC_DIAG_INPUT_DATATYPE
TC_DIAG_DATA_DOMAIN
TC_DIAG_DATA_LEVEL
TC_DIAG_REGRID_METHOD
TC_DIAG_REGRID_WIDTH
TC_DIAG_REGRID_VLD_THRESH
TC_DIAG_REGRID_SHAPE
TC_DIAG_REGRID_CENSOR_THRESH
TC_DIAG_REGRID_CENSOR_VAL
TC_DIAG_REGRID_CONVERT
TC_DIAG_COMPUTE_TANGENTIAL_AND_RADIAL_WINDS
TC_DIAG_U_WIND_FIELD_NAME
TC_DIAG_V_WIND_FIELD_NAME
TC_DIAG_TANGENTIAL_VELOCITY_FIELD_NAME
TC_DIAG_TANGENTIAL_VELOCITY_LONG_FIELD_NAME
TC_DIAG_RADIAL_VELOCITY_FIELD_NAME
TC_DIAG_RADIAL_VELOCITY_LONG_FIELD_NAME
TC_DIAG_VORTEX_REMOVAL
TC_DIAG_NC_RNG_AZI_FLAG
TC_DIAG_NC_DIAG_FLAG
TC_DIAG_CIRA_DIAG_FLAG
TC_DIAG_OUTPUT_PREFIX
TC_DIAG_SKIP_IF_OUTPUT_EXISTS
TC_DIAG_MET_CONFIG_OVERRIDES

6.28.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/TCDiagConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// TC-Diag configuration file.
//
// For additional information, please see the MET Users Guide.
//
////////////////////////////////////////////////////////////////////////////////

//
// Filter input track data lines.
//

//
// Model
//
//model =
${METPLUS_MODEL}


//
// Storm identifier
//
//storm_id =
${METPLUS_STORM_ID}


//
// Basin
//
//basin =
${METPLUS_BASIN}


//
// Cyclone number
//
//cyclone =
${METPLUS_CYCLONE}


//
// Model initialization time
//
//init_inc =
${METPLUS_INIT_INCLUDE}
```

(continues on next page)

(continued from previous page)

```
//
// Subset by the valid time
//
//valid_beg =
${METPLUS_VALID_BEG}
//valid_end =
${METPLUS_VALID_END}

//valid_inc =
${METPLUS_VALID_INCLUDE_LIST}

//valid_exc =
${METPLUS_VALID_EXCLUDE_LIST}


//
// Subset by the valid hour and lead time.
//
//valid_hour =
${METPLUS_VALID_HOUR_LIST}

//lead =
${METPLUS_LEAD_LIST}


////////////////////////////////////////////////////////////////////////////

//
// Python diagnostic scripts to be run
// May be set separately in each "domain_info" entry
//
//diag_script =
${METPLUS_DIAG_SCRIPT}

//
// Domain-specific cylindrical coordinate transformation
//
//domain_info = {
${METPLUS_DOMAIN_INFO_LIST}

////////////////////////////////////////////////////////////////////////////

//
// Data censoring and conversion
```

(continues on next page)

(continued from previous page)

```
// May be set separately in each diag_data "field" entry
//
// censor_thresh = [];
${METPLUS_CENSOR_THRESH}

// censor_val    = [];
${METPLUS_CENSOR_VAL}

// convert(x)    = x;
${METPLUS_CONVERT}


//
// Data fields
//
data = {

   ${METPLUS_DATA_FILE_TYPE}

   // If empty, the field is processed for all domains
   //domain = [];
   ${METPLUS_DATA_DOMAIN}

   // Pressure levels to be used, unless overridden below
   //level =
   ${METPLUS_DATA_LEVEL}

   //field = [
   ${METPLUS_DATA_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Regridding options
//
//regrid = {
${METPLUS_REGRID_DICT}

//
// Optionally convert u/v winds to tangential/radial winds
//
//compute_tangential_and_radial_winds =
${METPLUS_COMPUTE_TANGENTIAL_AND_RADIAL_WINDS}
```

(continues on next page)

(continued from previous page)

```
//u_wind_field_name =
${METPLUS_U_WIND_FIELD_NAME}

//v_wind_field_name =
${METPLUS_V_WIND_FIELD_NAME}

//tangential_velocity_field_name =
${METPLUS_TANGENTIAL_VELOCITY_FIELD_NAME}

//tangential_velocity_long_field_name =
${METPLUS_TANGENTIAL_VELOCITY_LONG_FIELD_NAME}

//radial_velocity_field_name =
${METPLUS_RADIAL_VELOCITY_FIELD_NAME}

//radial_velocity_long_field_name =
${METPLUS_RADIAL_VELOCITY_LONG_FIELD_NAME}


//
// Vortex removal flag
//
//vortex_removal =
${METPLUS_VORTEX_REMOVAL}


////////////////////////////////////////////////////////////////////////////////

//
// Flags to control output files
//
//nc_rng_azi_flag =
${METPLUS_NC_RNG_AZI_FLAG}

//nc_diag_flag =
${METPLUS_NC_DIAG_FLAG}

//cira_diag_flag =
${METPLUS_CIRA_DIAG_FLAG}


////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";
```

(continues on next page)

(continued from previous page)

```
//output_prefix =
${METPLUS_OUTPUT_PREFIX}

//version       = "V11.1.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_STORM_ID}

METplus Config(s)	MET Config File
TC_DIAG_STORM_ID	storm_id

${METPLUS_BASIN}

METplus Config(s)	MET Config File
TC_DIAG_BASIN	basin

${METPLUS_CYCLONE}

METplus Config(s)	MET Config File
TC_DIAG_CYCLONE	cyclone

${METPLUS_INIT_INCLUDE_LIST}

METplus Config(s)	MET Config File
TC_DIAG_INIT_INCLUDE	init_inc

${METPLUS_VALID_BEG}

METplus Config(s)	MET Config File
TC_DIAG_VALID_BEG	valid_beg

${METPLUS_VALID_END}

METplus Config(s)	MET Config File
TC_DIAG_VALID_END	valid_end

${METPLUS_VALID_INCLUDE_LIST}

METplus Config(s)	MET Config File
TC_DIAG_VALID_INCLUDE	valid_inc

${METPLUS_VALID_EXCLUDE_LIST}

METplus Config(s)	MET Config File
TC_DIAG_VALID_EXCLUDE	valid_exc

${METPLUS_VALID_HOUR_LIST}

METplus Config(s)	MET Config File
TC_DIAG_VALID_HOUR	valid_hour

${METPLUS_LEAD_LIST}

METplus Config(s)	MET Config File
LEAD_SEQ	lead

${METPLUS_DIAG_SCRIPT}

METplus Config(s)	MET Config File
TC_DIAG_DIAG_SCRIPT	diag_script

${METPLUS_DOMAIN_INFO_LIST}

METplus Config(s)	MET Config File
TC_DIAG_DOMAIN_INFO<n>_DOMAIN	domain_info.domain
TC_DIAG_DOMAIN_INFO<n>_N_RANGE	domain_info.n_range
TC_DIAG_DOMAIN_INFO<n>_N_AZIMUTH	domain_info.n_azimuth
TC_DIAG_DOMAIN_INFO<n>_DELTA_RANGE_KM	domain_info.delta_range_km
TC_DIAG_DOMAIN_INFO<n>_DIAG_SCRIPT	domain_info.diag_script

${METPLUS_CENSOR_THRESH}

METplus Config(s)	MET Config File
TC_DIAG_CENSOR_THRESH	censor_thresh

${METPLUS_CENSOR_VAL}

METplus Config(s)	MET Config File
TC_DIAG_CENSOR_VAL	censor_val

${METPLUS_CONVERT}

METplus Config(s)	MET Config File
TC_DIAG_CONVERT	convert

${METPLUS_DATA_FIELD}

METplus Config(s)	MET Config File
BOTH_VAR<n>_NAME	data.field.name
BOTH_VAR<n>_LEVELS	data.field.level
BOTH_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_DATA_FILE_TYPE}

METplus Config(s)	MET Config File
TC_DIAG_INPUT_DATATYPE	data.file_type

${METPLUS_DATA_DOMAIN}

METplus Config(s)	MET Config File
TC_DIAG_DATA_DOMAIN	data.domain

${METPLUS_DATA_LEVEL}

METplus Config(s)	MET Config File
TC_DIAG_DATA_LEVEL	data.level

${METPLUS_REGRID_DICT}

METplus Config(s)	MET Config File
TC_DIAG_REGRID_SHAPE	regrid.shape
TC_DIAG_REGRID_METHOD	regrid.method
TC_DIAG_REGRID_WIDTH	regrid.width
TC_DIAG_REGRID_VLD_THRESH	regrid.vld_thresh
TC_DIAG_REGRID_CONVERT	regrid.convert
TC_DIAG_REGRID_CENSOR_THRESH	regrid.censor_thresh
TC_DIAG_REGRID_CENSOR_VAL	regrid.censor_val

${METPLUS_COMPUTE_TANGENTIAL_AND_RADIAL_WINDS}

METplus Config(s)	MET Config File
TC_DIAG_COMPUTE_TANGENTIAL_AND_RADIAL_WINDS	compute_tangential_and_radial_winds

${METPLUS_U_WIND_FIELD_NAME}

METplus Config(s)	MET Config File
TC_DIAG_U_WIND_FIELD_NAME	u_wind_field_name

${METPLUS_V_WIND_FIELD_NAME}

METplus Config(s)	MET Config File
TC_DIAG_V_WIND_FIELD_NAME	v_wind_field_name

${METPLUS_TANGENTIAL_VELOCITY_FIELD_NAME}

METplus Config(s)	MET Config File
TC_DIAG_TANGENTIAL_VELOCITY_FIELD_NAME	tangential_velocity_field_name

${METPLUS_TANGENTIAL_VELOCITY_LONG_FIELD_NAME}

METplus Config(s)	MET Config File
TC_DIAG_TANGENTIAL_VELOCITY_LONG_FIELD_NAME	tangential_velocity_long_field_name

${METPLUS_RADIAL_VELOCITY_FIELD_NAME}

METplus Config(s)	MET Config File
TC_DIAG_RADIAL_VELOCITY_FIELD_NAME	radial_velocity_field_name

${METPLUS_RADIAL_VELOCITY_LONG_FIELD_NAME}

METplus Config(s)	MET Config File
TC_DIAG_RADIAL_VELOCITY_LONG_FIELD_NAME	radial_velocity_long_field_name

${METPLUS_VORTEX_REMOVAL}

METplus Config(s)	MET Config File
TC_DIAG_VORTEX_REMOVAL	vortex_removal

${METPLUS_NC_RNG_AZI_FLAG}

METplus Config(s)	MET Config File
TC_DIAG_NC_RNG_AZI_FLAG	nc_rng_azi_flag

${METPLUS_NC_DIAG_FLAG}

METplus Config(s)	MET Config File
TC_DIAG_NC_DIAG_FLAG	nc_diag_flag

${METPLUS_CIRA_DIAG_FLAG}

METplus Config(s)	MET Config File
TC_DIAG_CIRA_DIAG_FLAG	cira_diag_flag

${METPLUS_OUTPUT_PREFIX}

METplus Config(s)	MET Config File
TC_DIAG_OUTPUT_PREFIX	output_prefix

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
TC_DIAG_MET_CONFIG_OVERRIDES	n/a

6.29 TCGen

6.29.1 Description

The TCGen wrapper encapsulates the behavior of the MET tc_gen tool. The wrapper accepts track (Adeck or Bdeck) data and Genesis data.

6.29.2 METplus Configuration

TC_GEN_TRACK_INPUT_DIR
TC_GEN_TRACK_INPUT_TEMPLATE
TC_GEN_GENESIS_INPUT_DIR
TC_GEN_GENESIS_INPUT_TEMPLATE
TC_GEN_EDECK_INPUT_DIR
TC_GEN_EDECK_INPUT_TEMPLATE
TC_GEN_SHAPE_INPUT_DIR
TC_GEN_SHAPE_INPUT_TEMPLATE
TC_GEN_OUTPUT_DIR
TC_GEN_OUTPUT_TEMPLATE
LOG_TC_GEN_VERBOSITY
TC_GEN_CUSTOM_LOOP_LIST
TC_GEN_SKIP_IF_OUTPUT_EXISTS

TC_GEN_MET_CONFIG_OVERRIDES
TC_GEN_CONFIG_FILE
TC_GEN_INIT_FREQ
TC_GEN_VALID_FREQ
TC_GEN_FCST_HR_WINDOW_BEGIN
TC_GEN_FCST_HR_WINDOW_END
TC_GEN_MIN_DURATION
TC_GEN_FCST_GENESIS_VMAX_THRESH
TC_GEN_FCST_GENESIS_MSLP_THRESH
TC_GEN_BEST_GENESIS_TECHNIQUE
TC_GEN_BEST_GENESIS_CATEGORY
TC_GEN_BEST_GENESIS_VMAX_THRESH
TC_GEN_BEST_GENESIS_MSLP_THRESH
TC_GEN_OPER_TECHNIQUE
TC_GEN_FILTER_<n>
TC_GEN_DESC
MODEL
TC_GEN_STORM_ID
TC_GEN_STORM_NAME
TC_GEN_INIT_BEG
TC_GEN_INIT_END
TC_GEN_INIT_INC
TC_GEN_INIT_EXC
TC_GEN_VALID_BEG
TC_GEN_VALID_END
TC_GEN_INIT_HOUR
LEAD_SEQ
TC_GEN_VX_MASK
TC_GEN_BASIN_MASK
TC_GEN_DLAND_THRESH
TC_GEN_GENESIS_MATCH_RADIUS
TC_GEN_GENESIS_MATCH_POINT_TO_TRACK
TC_GEN_GENESIS_MATCH_WINDOW_BEG
TC_GEN_GENESIS_MATCH_WINDOW_END
TC_GEN_DEV_HIT_RADIUS
TC_GEN_DEV_HIT_WINDOW_BEGIN
TC_GEN_DEV_HIT_WINDOW_END
TC_GEN_OPS_HIT_WINDOW_BEG
TC_GEN_OPS_HIT_WINDOW_END
TC_GEN_DISCARD_INIT_POST_GENESIS_FLAG
TC_GEN_DEV_METHOD_FLAG

TC_GEN_OPS_METHOD_FLAG
TC_GEN_CI_ALPHA
TC_GEN_OUTPUT_FLAG_FHO
TC_GEN_OUTPUT_FLAG_CTC
TC_GEN_OUTPUT_FLAG_CTS
TC_GEN_OUTPUT_FLAG_PCT
TC_GEN_OUTPUT_FLAG_PSTD
TC_GEN_OUTPUT_FLAG_PJC
TC_GEN_OUTPUT_FLAG_PRC
TC_GEN_OUTPUT_FLAG_GENMPR
TC_GEN_NC_PAIRS_FLAG_LATLON
TC_GEN_NC_PAIRS_FLAG_FCST_GENESIS
TC_GEN_NC_PAIRS_FLAG_FCST_TRACKS
TC_GEN_NC_PAIRS_FLAG_FCST_FY_OY
TC_GEN_NC_PAIRS_FLAG_FCST_FY_ON
TC_GEN_NC_PAIRS_FLAG_BEST_GENESIS
TC_GEN_NC_PAIRS_FLAG_BEST_TRACKS
TC_GEN_NC_PAIRS_FLAG_BEST_FY_OY
TC_GEN_NC_PAIRS_FLAG_BEST_FN_OY
TC_GEN_VALID_MINUS_GENESIS_DIFF_THRESH
TC_GEN_BEST_UNIQUE_FLAG
TC_GEN_DLAND_FILE
TC_GEN_BASIN_FILE
TC_GEN_NC_PAIRS_GRID

Warning: DEPRECATED:

TC_GEN_LEAD_WINDOW_BEGIN
TC_GEN_LEAD_WINDOW_END
TC_GEN_OPER_GENESIS_TECHNIQUE
TC_GEN_OPER_GENESIS_CATEGORY
TC_GEN_OPER_GENESIS_VMAX_THRESH
TC_GEN_OPER_GENESIS_MSLP_THRESH
TC_GEN_GENESIS_RADIUS
TC_GEN_GENESIS_WINDOW_BEGIN
TC_GEN_GENESIS_WINDOW_END

6.29.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/TCGenConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// TC-Gen configuration file.
//
// For additional information, see the MET_BASE/config/README_TC file.
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

////////////////////////////////////////////////////////////////////////////////
//
// Genesis event definition criteria.
//
////////////////////////////////////////////////////////////////////////////////

//
// Model initialization frequency in hours, starting at 0.
//
// init_freq =
${METPLUS_INIT_FREQ}

//
// Valid hour frequency to be analyzed in hours, starting at 0
//
// valid_freq =
${METPLUS_VALID_FREQ}

//
// Forecast hours to be searched for genesis events
//
// fcst_hr_window =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_FCST_HR_WINDOW_DICT}

//
// Minimum track duration for genesis event in hours.
//
// min_duration =
${METPLUS_MIN_DURATION}

//
// Forecast genesis event criteria.  Defined as tracks reaching the specified
// intensity category, maximum wind speed threshold, and minimum sea-level
// pressure threshold.  The forecast genesis time is the valid time of the first
// track point where all of these criteria are met.
//
// fcst_genesis =
${METPLUS_FCST_GENESIS_DICT}

//
// BEST track genesis event criteria.  Defined as tracks reaching the specified
// intensity category, maximum wind speed threshold, and minimum sea-level
// pressure threshold.  The BEST track genesis time is the valid time of the
// first track point where all of these criteria are met.
//
// best_genesis =
${METPLUS_BEST_GENESIS_DICT}

//
// Operational track technique name
//
// oper_technique =
${METPLUS_OPER_TECHNIQUE}

////////////////////////////////////////////////////////////////////////////////
//
// Track filtering options
// May be specified separately in each filter array entry.
//
////////////////////////////////////////////////////////////////////////////////

//
// Array of dictionaries containing the track filtering options
// If empty, a single filter is defined using the top-level settings.
//
// filter =
${METPLUS_FILTER}
```

(continues on next page)

(continued from previous page)

```
//
// Description written to output DESC column
//
// desc =
${METPLUS_DESC}

//
// Forecast ATCF ID's
// If empty, all ATCF ID's found will be processed.
// Statistics will be generated separately for each ATCF ID.
//
// model =
${METPLUS_MODEL}

//
// BEST and operational track storm identifiers
//
// storm_id =
${METPLUS_STORM_ID}

//
// BEST and operational track storm names
//
// storm_name =
${METPLUS_STORM_NAME}

//
// Forecast and operational initialization times to include or exclude
//
// init_beg =
${METPLUS_INIT_BEG}

// init_end =
${METPLUS_INIT_END}

// init_inc =
${METPLUS_INIT_INC}

// init_exc =
${METPLUS_INIT_EXC}

//
// Forecast, BEST, and operational valid time window
//
```

(continues on next page)

(continued from previous page)

```
// valid_beg =
${METPLUS_VALID_BEG}

// valid_end =
${METPLUS_VALID_END}

//
// Forecast and operational initialization hours
//
// init_hour =
${METPLUS_INIT_HOUR}

//
// Forecast and operational lead times in hours
//
// lead =
${METPLUS_LEAD}

//
// Spatial masking region (path to gridded data file or polyline file)
//
// vx_mask =
${METPLUS_VX_MASK}

//
// Spatial masking of hurricane basin names from the basin_file
//
// basin_mask =
${METPLUS_BASIN_MASK}

//
// Distance to land threshold
//
//dland_thresh =
${METPLUS_DLAND_THRESH}

////////////////////////////////////////////////////////////////////////////////
//
// Matching and scoring options
// May be specified separately in each filter array entry.
//
////////////////////////////////////////////////////////////////////////////////

//
// Genesis matching logic. Compare the forecast genesis point to all points in
```

(continues on next page)

(continued from previous page)

```
// the Best track (TRUE) or the single Best track genesis point (FALSE).
//
//genesis_match_point_to_track =
${METPLUS_GENESIS_MATCH_POINT_TO_TRACK}

//
// Radius in km to search for a matching genesis event
//
// genesis_match_radius =
${METPLUS_GENESIS_MATCH_RADIUS}

//
// Time window in hours, relative to the model genesis time, to search for a
// matching Best track point
//
//genesis_match_window = {
${METPLUS_GENESIS_MATCH_WINDOW_DICT}

//
// Radius in km for a development scoring method hit
//
// dev_hit_radius =
${METPLUS_DEV_HIT_RADIUS}

//
// Time window in hours for a development scoring method hit
//
// dev_hit_window =
${METPLUS_DEV_HIT_WINDOW_DICT}

// Time window in hours for the Best track genesis minus model initialization
// time difference for an operational scoring method hit
//
//ops_hit_window = {
${METPLUS_OPS_HIT_WINDOW_DICT}

//
// Discard genesis forecasts for initializations at or after the matching
// BEST track genesis time
//
// discard_init_post_genesis_flag =
${METPLUS_DISCARD_INIT_POST_GENESIS_FLAG}

//
// Scoring methods to be applied
```

(continues on next page)

(continued from previous page)

```
//
//dev_method_flag =
${METPLUS_DEV_METHOD_FLAG}

// ops_method_flag =
${METPLUS_OPS_METHOD_FLAG}

////////////////////////////////////////////////////////////////////////////////
//
// Output options
// May be specified separately in each filter array entry.
//
////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval alpha value
//
// ci_alpha =
${METPLUS_CI_ALPHA}

//
// Statistical output types
//
// output_flag =
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF genesis pair counts
//
// nc_pairs_flag =
${METPLUS_NC_PAIRS_FLAG_DICT}

//
// Specify which track points should be counted by thresholding the track point
// valid time minus genesis time difference.
//
// valid_minus_genesis_diff_thresh =
${METPLUS_VALID_MINUS_GENESIS_DIFF_THRESH}

//
// Count unique BEST track genesis event locations (TRUE) versus counting the
// location for all pairs (FALSE).
//
// best_unique_flag =
${METPLUS_BEST_UNIQUE_FLAG}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////
//
// Global settings
// May only be specified once.
//
////////////////////////////////////////////////////////////////////////////////

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
// dland_file =
${METPLUS_DLAND_FILE}

//
// Specify the NetCDF file containing a gridded representation of the
// global basins.
//
// basin_file =
${METPLUS_BASIN_FILE}

//
// NetCDF genesis pairs grid
//
// nc_pairs_grid =
${METPLUS_NC_PAIRS_GRID}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V10.0.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_INIT_FREQ}

METplus Config(s)	MET Config File
TC_GEN_INIT_FREQ	init_freq

${METPLUS_VALID_FREQ}

METplus Config(s)	MET Config File
TC_GEN_VALID_FREQ	valid_freq

${METPLUS_FCST_HR_WINDOW_DICT}

METplus Config(s)	MET Config File
TC_GEN_FCST_HR_WINDOW_BEGIN	fcst_hr_window.beg
TC_GEN_FCST_HR_WINDOW_END	fcst_hr_window.end

${METPLUS_MIN_DURATION}

METplus Config(s)	MET Config File
TC_GEN_MIN_DURATION	min_duration

${METPLUS_FCST_GENESIS_DICT}

METplus Config(s)	MET Config File
TC_GEN_FCST_GENESIS_VMAX_THRESH	fcst_genesis.vmax_thresh
TC_GEN_FCST_GENESIS_MSLP_THRESH	fcst_genesis.mslp_thresh

${METPLUS_BEST_GENESIS_DICT}

METplus Config(s)	MET Config File
TC_GEN_BEST_GENESIS_TECHNIQUE	best_genesis.technique
TC_GEN_BEST_GENESIS_CATEGORY	best_genesis.category
TC_GEN_BEST_GENESIS_VMAX_THRESH	best_genesis.vmax_thresh
TC_GEN_BEST_GENESIS_MSLP_THRESH	best_genesis.mslp_thresh

${METPLUS_OPER_TECHNIQUE}

METplus Config(s)	MET Config File
TC_GEN_OPER_TECHNIQUE	oper_technique

${METPLUS_FILTER}

METplus Config(s)	MET Config File
TC_GEN_FILTER_<n>	filter

${METPLUS_DESC}

METplus Config(s)	MET Config File
DESC -or- *TC_GEN_DESC*	desc

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_STORM_ID}

METplus Config(s)	MET Config File
TC_GEN_STORM_ID	storm_id

${METPLUS_STORM_NAME}

METplus Config(s)	MET Config File
TC_GEN_STORM_NAME	storm_name

${METPLUS_INIT_BEG}

METplus Config(s)	MET Config File
TC_GEN_INIT_BEG	init_beg

${METPLUS_INIT_END}

METplus Config(s)	MET Config File
TC_GEN_INIT_END	init_end

${METPLUS_INIT_INC}

METplus Config(s)	MET Config File
TC_GEN_INIT_INC	init_inc

${METPLUS_INIT_EXC}

METplus Config(s)	MET Config File
TC_GEN_INIT_EXC	init_exc

${METPLUS_VALID_BEG}

METplus Config(s)	MET Config File
TC_GEN_VALID_BEG	valid_beg

${METPLUS_VALID_END}

METplus Config(s)	MET Config File
TC_GEN_VALID_END	valid_end

${METPLUS_INIT_HOUR}

METplus Config(s)	MET Config File
TC_GEN_INIT_HOUR	init_hour

${METPLUS_LEAD}

METplus Config(s)	MET Config File
LEAD_SEQ	lead

${METPLUS_VX_MASK}

METplus Config(s)	MET Config File
TC_GEN_VX_MASK	vx_mask

${METPLUS_BASIN_MASK}

METplus Config(s)	MET Config File
TC_GEN_BASIN_MASK	basin_mask

${METPLUS_DLAND_THRESH}

METplus Config(s)	MET Config File
TC_GEN_DLAND_THRESH	dland_thresh

${METPLUS_DEV_HIT_WINDOW_DICT}

METplus Config(s)	MET Config File
TC_GEN_DEV_HIT_WINDOW_BEGIN	dev_hit_window.beg
TC_GEN_DEV_HIT_WINDOW_END	dev_hit_window.end

${METPLUS_GENESIS_MATCH_RADIUS}

METplus Config(s)	MET Config File
TC_GEN_GENESIS_MATCH_RADIUS	genesis_match_radius

${METPLUS_GENESIS_MATCH_POINT_TO_TRACK}

METplus Config(s)	MET Config File
TC_GEN_GENESIS_MATCH_POINT_TO_TRACK	genesis_match_point_to_track

${METPLUS_GENESIS_MATCH_WINDOW_DICT}

METplus Config(s)	MET Config File
TC_GEN_GENESIS_MATCH_WINDOW_BEG	genesis_match_window.beg
TC_GEN_GENESIS_MATCH_WINDOW_END	genesis_match_window.end

${METPLUS_DEV_HIT_RADIUS}

METplus Config(s)	MET Config File
TC_GEN_DEV_HIT_RADIUS	dev_hit_radius

${METPLUS_OPS_HIT_WINDOW_DICT}

METplus Config(s)	MET Config File
TC_GEN_OPS_HIT_WINDOW_BEG	ops_hit_window.beg
TC_GEN_OPS_HIT_WINDOW_END	ops_hit_window.end

${METPLUS_DISCARD_INIT_POST_GENESIS_FLAG}

METplus Config(s)	MET Config File
TC_GEN_DISCARD_INIT_POST_GENESIS_FLAG	discard_init_post_genesis_flag

${METPLUS_DEV_METHOD_FLAG}

METplus Config(s)	MET Config File
TC_GEN_DEV_METHOD_FLAG	dev_method_flag

${METPLUS_OPS_METHOD_FLAG}

METplus Config(s)	MET Config File
TC_GEN_OPS_METHOD_FLAG	ops_method_flag

${METPLUS_CI_ALPHA}

METplus Config(s)	MET Config File
TC_GEN_CI_ALPHA	ci_alpha

${METPLUS_OUTPUT_FLAG_DICT}

METplus Config(s)	MET Config File
TC_GEN_OUTPUT_FLAG_FHO	output_flag.fho
TC_GEN_OUTPUT_FLAG_CTC	output_flag.ctc
TC_GEN_OUTPUT_FLAG_CTS	output_flag.cts
TC_GEN_OUTPUT_FLAG_PCT	output_flag.pct
TC_GEN_OUTPUT_FLAG_PSTD	output_flag.pstd
TC_GEN_OUTPUT_FLAG_PJC	output_flag.pjc
TC_GEN_OUTPUT_FLAG_PRC	output_flag.prc
TC_GEN_OUTPUT_FLAG_GENMPR	output_flag.genmpr

${METPLUS_NC_PAIRS_FLAG_DICT}

METplus Config(s)	MET Config File
TC_GEN_NC_PAIRS_FLAG_LATLON	nc_pairs_flag.latlon
TC_GEN_NC_PAIRS_FLAG_FCST_GENESIS	nc_pairs_flag.fcst_genesis
TC_GEN_NC_PAIRS_FLAG_FCST_TRACKS	nc_pairs_flag.fcst_tracks
TC_GEN_NC_PAIRS_FLAG_FCST_FY_OY	nc_pairs_flag.fcst_fy_oy
TC_GEN_NC_PAIRS_FLAG_FCST_FY_ON	nc_pairs_flag.fcst_fy_on
TC_GEN_NC_PAIRS_FLAG_BEST_GENESIS	nc_pairs_flag.best_genesis
TC_GEN_NC_PAIRS_FLAG_BEST_TRACKS	nc_pairs_flag.best_tracks
TC_GEN_NC_PAIRS_FLAG_BEST_FY_OY	nc_pairs_flag.best_fy_oy
TC_GEN_NC_PAIRS_FLAG_BEST_FN_OY	nc_pairs_flag.best_fn_oy

${METPLUS_VALID_MINUS_GENESIS_DIFF_THRESH}

METplus Config(s)	MET Config File
TC_GEN_VALID_MINUS_GENESIS_DIFF_THRESH	valid_minus_genesis_diff_thresh

${METPLUS_BEST_UNIQUE_FLAG}

METplus Config(s)	MET Config File
TC_GEN_BEST_UNIQUE_FLAG	best_unique_flag

${METPLUS_DLAND_FILE}

METplus Config(s)	MET Config File
TC_GEN_DLAND_FILE	dland_file

${METPLUS_BASIN_FILE}

METplus Config(s)	MET Config File
TC_GEN_BASIN_FILE	basin_file

${METPLUS_NC_PAIRS_GRID}

METplus Config(s)	MET Config File
TC_GEN_NC_PAIRS_GRID	nc_pairs_grid

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
TC_GEN_MET_CONFIG_OVERRIDES	n/a

6.30 TCMPRPlotter

6.30.1 Description

The TCMPRPlotter wrapper is a Python script that wraps the R script plot_tcmpr.R. This script is useful for plotting the calculated statistics for the output from the MET-TC tools. This script, and other R scripts are included in the MET installation. Please refer to the MET User's Guide for usage information.

6.30.2 METplus Configuration

TCMPR_PLOTTER_TCMPR_DATA_DIR
TCMPR_PLOTTER_PLOT_OUTPUT_DIR
TCMPR_PLOTTER_CONFIG_FILE
TCMPR_PLOTTER_PREFIX
TCMPR_PLOTTER_TITLE
TCMPR_PLOTTER_SUBTITLE
TCMPR_PLOTTER_XLAB
TCMPR_PLOTTER_YLAB
TCMPR_PLOTTER_XLIM
TCMPR_PLOTTER_YLIM
TCMPR_PLOTTER_FILTER
TCMPR_PLOTTER_FILTERED_TCST_DATA_FILE
TCMPR_PLOTTER_DEP_VARS
TCMPR_PLOTTER_SCATTER_X
TCMPR_PLOTTER_SCATTER_Y
TCMPR_PLOTTER_SKILL_REF
TCMPR_PLOTTER_SERIES
TCMPR_PLOTTER_SERIES_CI
TCMPR_PLOTTER_LEGEND
TCMPR_PLOTTER_LEAD
TCMPR_PLOTTER_PLOT_TYPES
TCMPR_PLOTTER_RP_DIFF
TCMPR_PLOTTER_DEMO_YR

TCMPR_PLOTTER_HFIP_BASELINE
TCMPR_PLOTTER_FOOTNOTE_FLAG
TCMPR_PLOTTER_PLOT_CONFIG_OPTS
TCMPR_PLOTTER_SAVE_DATA
TCMPR_PLOTTER_DEP_LABELS
TCMPR_PLOTTER_PLOT_LABELS
TCMPR_PLOTTER_READ_ALL_FILES

The following are TCMPR flags, if set to 'no', then don't set flag, if set to 'yes', then set the flag

TCMPR_PLOTTER_NO_EE
TCMPR_PLOTTER_NO_LOG
TCMPR_PLOTTER_SAVE

Warning: DEPRECATED:

TCMPR_PLOT_OUT_DIR
TITLE
SUBTITLE
XLAB
YLAB
XLIM
YLIM
FILTER
FILTERED_TCST_DATA_FILE
DEP_VARS
SCATTER_X
SCATTER_Y
SKILL_REF
SERIES
SERIES_CI
LEGEND
LEAD
PLOT_TYPES
RP_DIFF

DEMO_YR
HFIP_BASELINE
FOOTNOTE_FLAG
PLOT_CONFIG_OPTS
SAVE_DATA

6.31 TCPairs

6.31.1 Description

The TCPairs wrapper encapsulates the behavior of the MET tc_pairs tool. The wrapper accepts Adeck and Bdeck (Best track) cyclone track data in extra tropical cyclone format (such as the data used by sample data provided in the METplus tutorial), or ATCF formatted track data. If data is in an extra tropical cyclone (non-ATCF) format, the data is reformatted into an ATCF format that is recognized by MET.

6.31.2 METplus Configuration

TC_PAIRS_ADECK_INPUT_DIR
TC_PAIRS_BDECK_INPUT_DIR
TC_PAIRS_EDECK_INPUT_DIR
TC_PAIRS_OUTPUT_DIR
TC_PAIRS_REFORMAT_DIR
TC_PAIRS_ADECK_INPUT_TEMPLATE
TC_PAIRS_BDECK_INPUT_TEMPLATE
TC_PAIRS_EDECK_INPUT_TEMPLATE
TC_PAIRS_OUTPUT_TEMPLATE
TC_PAIRS_CONFIG_FILE
TC_PAIRS_INIT_INCLUDE
TC_PAIRS_INIT_EXCLUDE
TC_PAIRS_VALID_INCLUDE
TC_PAIRS_VALID_EXCLUDE
TC_PAIRS_WRITE_VALID
TC_PAIRS_READ_ALL_FILES
TC_PAIRS_MODEL
TC_PAIRS_STORM_ID
TC_PAIRS_BASIN
TC_PAIRS_CYCLONE
TC_PAIRS_STORM_NAME
TC_PAIRS_DLAND_FILE

TC_PAIRS_MISSING_VAL_TO_REPLACE
TC_PAIRS_MISSING_VAL
TC_PAIRS_SKIP_IF_REFORMAT_EXISTS
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS
TC_PAIRS_REFORMAT_DECK
TC_PAIRS_REFORMAT_TYPE
TC_PAIRS_CUSTOM_LOOP_LIST
TC_PAIRS_DESC
TC_PAIRS_MET_CONFIG_OVERRIDES
TC_PAIRS_CONSENSUS<n>_NAME
TC_PAIRS_CONSENSUS<n>_MEMBERS
TC_PAIRS_CONSENSUS<n>_REQUIRED
TC_PAIRS_CONSENSUS<n>_MIN_REQ
TC_PAIRS_CONSENSUS<n>_WRITE_MEMBERS
TC_PAIRS_SKIP_LEAD_SEQ
TC_PAIRS_RUN_ONCE
TC_PAIRS_CHECK_DUP
TC_PAIRS_INTERP12
TC_PAIRS_MATCH_POINTS
TC_PAIRS_DIAG_INFO_MAP<n>_DIAG_SOURCE
TC_PAIRS_DIAG_INFO_MAP<n>_TRACK_SOURCE
TC_PAIRS_DIAG_INFO_MAP<n>_FIELD_SOURCE
TC_PAIRS_DIAG_INFO_MAP<n>_MATCH_TO_TRACK
TC_PAIRS_DIAG_INFO_MAP<n>_DIAG_NAME
TC_PAIRS_DIAG_CONVERT_MAP<n>_DIAG_SOURCE
TC_PAIRS_DIAG_CONVERT_MAP<n>_KEY
TC_PAIRS_DIAG_CONVERT_MAP<n>_CONVERT
TC_PAIRS_DIAG_SOURCE<n>
TC_PAIRS_DIAG_TEMPLATE<n>
TC_PAIRS_DIAG_DIR<n>

Warning: DEPRECATED:

ADECK_TRACK_DATA_DIR
BDECK_TRACK_DATA_DIR
TRACK_DATA_SUBDIR_MOD
TC_PAIRS_DIR
TOP_LEVEL_DIRS

MODEL
STORM_ID
BASIN
CYCLONE
STORM_NAME
DLAND_FILE
TRACK_TYPE
ADECK_FILE_PREFIX
BDECK_FILE_PREFIX
MISSING_VAL_TO_REPLACE
MISSING_VAL
INIT_INCLUDE
INIT_EXCLUDE
INIT_HOUR_END

6.31.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/TCPairsConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

//
// Models
//
${METPLUS_MODEL}
```

(continues on next page)

(continued from previous page)

```
//
// Description
//
${METPLUS_DESC}

//
// Storm identifiers
//
${METPLUS_STORM_ID}

//
// Basins
//
${METPLUS_BASIN}

//
// Cyclone numbers
//
${METPLUS_CYCLONE}

//
// Storm names
//
${METPLUS_STORM_NAME}

//
// Model initialization time windows to include or exclude
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
${METPLUS_VALID_INC}
// valid_exc =
${METPLUS_VALID_EXC}

// write_valid =
${METPLUS_WRITE_VALID}

//
```

(continues on next page)

(continued from previous page)

```
// Valid model time window
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}

//
// Model initialization hours
//
init_hour = [];

//
// Required lead time in hours
//
lead_req = [];

//
// lat/lon polylines defining masking regions
//
init_mask  = "";
valid_mask = "";

//
// Specify if the code should check for duplicate ATCF lines
//
//check_dup =
${METPLUS_CHECK_DUP}


//
// Specify special processing to be performed for interpolated models.
// Set to NONE, FILL, or REPLACE.
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
//
//consensus =
${METPLUS_CONSENSUS_LIST}


//
// Forecast lag times
//
```

(continues on next page)

(continued from previous page)

```
lag_time = [];

//
// CLIPER/SHIFOR baseline forecasts to be derived from the BEST
// and operational (CARQ) tracks.
//
best_technique = [ "BEST" ];
best_baseline  = [];
oper_technique = [ "CARQ" ];
oper_baseline  = [];

//
// Specify the datasets to be searched for analysis tracks (NONE, ADECK, BDECK,
// or BOTH).
//
anly_track = BDECK;

//
// Specify if only those track points common to both the ADECK and BDECK
// tracks be written out.
//
//match_points =
${METPLUS_MATCH_POINTS}

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
${METPLUS_DLAND_FILE}

//
// Specify watch/warning information:
//   - Input watch/warning filename
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}


//diag_info_map = {
${METPLUS_DIAG_INFO_MAP_LIST}

//diag_convert_map = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_DIAG_CONVERT_MAP_LIST}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_DESC}

METplus Config(s)	MET Config File
DESC -or- *TC_PAIRS_DESC*	desc

${METPLUS_STORM_ID}

METplus Config(s)	MET Config File
TC_PAIRS_STORM_ID	storm_id

${METPLUS_BASIN}

METplus Config(s)	MET Config File
TC_PAIRS_BASIN	basin

${METPLUS_CYCLONE}

METplus Config(s)	MET Config File
TC_PAIRS_CYCLONE	cyclone

${METPLUS_STORM_NAME}

METplus Config(s)	MET Config File
TC_PAIRS_STORM_NAME	storm_name

${METPLUS_INIT_BEG}

METplus Config(s)	MET Config File
TC_PAIRS_INIT_BEG	init_beg

${METPLUS_INIT_END}

METplus Config(s)	MET Config File
TC_PAIRS_INIT_END	init_end

${METPLUS_INIT_INC}

METplus Config(s)	MET Config File
TC_PAIRS_INIT_INCLUDE	init_inc

${METPLUS_INIT_EXC}

METplus Config(s)	MET Config File
TC_PAIRS_INIT_EXCLUDE	init_exc

${METPLUS_VALID_INC}

METplus Config(s)	MET Config File
TC_PAIRS_VALID_INCLUDE	valid_inc

${METPLUS_VALID_EXC}

METplus Config(s)	MET Config File
TC_PAIRS_VALID_EXCLUDE	valid_exc

${METPLUS_WRITE_VALID}

METplus Config(s)	MET Config File
TC_PAIRS_WRITE_VALID	write_valid

${METPLUS_VALID_BEG}

METplus Config(s)	MET Config File
TC_PAIRS_VALID_BEG	valid_beg

${METPLUS_VALID_END}

METplus Config(s)	MET Config File
TC_PAIRS_VALID_END	valid_end

${METPLUS_MATCH_POINTS}

METplus Config(s)	MET Config File
TC_PAIRS_MATCH_POINTS	match_points

${METPLUS_DLAND_FILE}

METplus Config(s)	MET Config File
TC_PAIRS_DLAND_FILE	dland_file

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
TC_PAIRS_MET_CONFIG_OVERRIDES	n/a

${METPLUS_CONSENSUS_LIST}

METplus Config(s)	MET Config File
TC_PAIRS_CONSENSUS<n>_NAME	consensus.name
TC_PAIRS_CONSENSUS<n>_MEMBERS	consensus.members
TC_PAIRS_CONSENSUS<n>_REQUIRED	consensus.required
TC_PAIRS_CONSENSUS<n>_MIN_REQ	consensus.min_req
TC_PAIRS_CONSENSUS<n>_WRITE_MEMBERS	consensus.write_members

${METPLUS_CHECK_DUP}

METplus Config(s)	MET Config File
TC_PAIRS_CHECK_DUP	check_dup

${METPLUS_INTERP12}

METplus Config(s)	MET Config File
TC_PAIRS_INTERP12	interp12

${METPLUS_DIAG_INFO_MAP_LIST}

METplus Config(s)	MET Config File
TC_PAIRS_DIAG_INFO_MAP<n>_DIAG_SOURCE	diag_info_map.diag_source
TC_PAIRS_DIAG_INFO_MAP<n>_TRACK_SOURCE	diag_info_map.track_source
TC_PAIRS_DIAG_INFO_MAP<n>_FIELD_SOURCE	diag_info_map.field_source
TC_PAIRS_DIAG_INFO_MAP<n>_MATCH_TO_TRACK	diag_info_map.match_to_track
TC_PAIRS_DIAG_INFO_MAP<n>_DIAG_NAME	diag_info_map.diag_name

${METPLUS_DIAG_CONVERT_MAP_LIST}

METplus Config(s)	MET Config File
TC_PAIRS_DIAG_CONVERT_MAP<n>_DIAG_SOURCE	diag_convert_map.diag_source
TC_PAIRS_DIAG_CONVERT_MAP<n>_KEY	diag_convert_map.key
TC_PAIRS_DIAG_CONVERT_MAP<n>_CONVERT	diag_convert_map.convert

6.32 TCRMW

6.32.1 Description

Used to configure the MET tool TC-RMW.

6.32.2 METplus Configuration

TC_RMW_INPUT_DIR
TC_RMW_DECK_INPUT_DIR
TC_RMW_OUTPUT_DIR
TC_RMW_DECK_TEMPLATE
TC_RMW_INPUT_TEMPLATE
TC_RMW_INPUT_FILE_LIST
TC_RMW_OUTPUT_TEMPLATE
LOG_TC_RMW_VERBOSITY
TC_RMW_CONFIG_FILE
TC_RMW_INPUT_DATATYPE
TC_RMW_REGRID_METHOD
TC_RMW_REGRID_WIDTH
TC_RMW_REGRID_VLD_THRESH
TC_RMW_REGRID_SHAPE
TC_RMW_REGRID_CONVERT
TC_RMW_REGRID_CENSOR_THRESH
TC_RMW_REGRID_CENSOR_VAL
TC_RMW_N_RANGE
TC_RMW_N_AZIMUTH
TC_RMW_MAX_RANGE_KM
TC_RMW_DELTA_RANGE_KM
TC_RMW_SCALE
TC_RMW_STORM_ID
TC_RMW_BASIN
TC_RMW_CYCLONE
TC_RMW_STORM_NAME
TC_RMW_INIT_INCLUDE

TC_RMW_VALID_BEG
TC_RMW_VALID_END
TC_RMW_VALID_INCLUDE_LIST
TC_RMW_VALID_EXCLUDE_LIST
TC_RMW_VALID_HOUR_LIST
TC_RMW_SKIP_IF_OUTPUT_EXISTS
TC_RMW_DESC
MODEL
LEAD_SEQ
TC_RMW_MET_CONFIG_OVERRIDES

6.32.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/TCRMWConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// TC-RMW configuration file.
//
// For additional information, see the MET_BASE/config/README_TC file.
//
////////////////////////////////////////////////////////////////////////////////

// The following environment variables set the text if the corresponding
// variables at defined in the METplus config. If not, they are set to
// and empty string, which will cause MET to use the value defined in the
// default configuration file.

${METPLUS_MODEL}

${METPLUS_STORM_ID}
${METPLUS_BASIN}
${METPLUS_CYCLONE}
${METPLUS_INIT_INCLUDE}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INCLUDE_LIST}
${METPLUS_VALID_EXCLUDE_LIST}

${METPLUS_VALID_HOUR_LIST}
${METPLUS_LEAD_LIST}

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];

//
// Data fields
//
data = {
   ${METPLUS_DATA_FILE_TYPE}

   ${METPLUS_DATA_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Regridding options
//
${METPLUS_REGRID_DICT}

//
// Range-Azimuth grid parameters
//
// The following environmnet variables set the text if the corresponding
// variables at defined in the METplus config. If not, they are set to
// and empty string, which will cause MET to use the value defined in the
// default configuration file.

${METPLUS_N_RANGE}
${METPLUS_N_AZIMUTH}
${METPLUS_MAX_RANGE_KM}
${METPLUS_DELTA_RANGE_KM}
${METPLUS_RMW_SCALE}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//version = "V10.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_MODEL}

METplus Config(s)	MET Config File
MODEL	model

${METPLUS_STORM_ID}

METplus Config(s)	MET Config File
TC_RMW_STORM_ID	storm_id

${METPLUS_BASIN}

METplus Config(s)	MET Config File
TC_RMW_BASIN	basin

${METPLUS_CYCLONE}

METplus Config(s)	MET Config File
TC_RMW_CYCLONE	cyclone

${METPLUS_INIT_INCLUDE}

METplus Config(s)	MET Config File
TC_RMW_INIT_INCLUDE	init_inc

${METPLUS_VALID_BEG}

METplus Config(s)	MET Config File
TC_RMW_VALID_BEG	valid_beg

${METPLUS_VALID_END}

METplus Config(s)	MET Config File
TC_RMW_VALID_END	valid_end

${METPLUS_VALID_INCLUDE_LIST}

METplus Config(s)	MET Config File
TC_RMW_VALID_INCLUDE_LIST	valid_inc

${METPLUS_VALID_EXCLUDE_LIST}

METplus Config(s)	MET Config File
TC_RMW_VALID_EXCLUDE_LIST	valid_exc

${METPLUS_VALID_HOUR_LIST}

METplus Config(s)	MET Config File
TC_RMW_VALID_HOUR_LIST	valid_hour

${METPLUS_LEAD_LIST}

METplus Config(s)	MET Config File
LEAD_SEQ	lead

${METPLUS_DATA_FILE_TYPE}

METplus Config(s)	MET Config File
TC_RMW_INPUT_DATATYPE	data.file_type

${METPLUS_DATA_FIELD}

METplus Config(s)	MET Config File
BOTH_VAR<n>_NAME	data.field.name
BOTH_VAR<n>_LEVELS	data.field.level
BOTH_VAR<n>_OPTIONS	n/a

Note: For more information on controlling the field attributes in METplus, please see the *Field Info* (page 58) section of the User's Guide.

${METPLUS_REGRID_DICT}

METplus Config(s)	MET Config File
TC_RMW_REGRID_SHAPE	regrid.shape
TC_RMW_REGRID_METHOD	regrid.method
TC_RMW_REGRID_WIDTH	regrid.width
TC_RMW_REGRID_VLD_THRESH	regrid.vld_thresh
TC_RMW_REGRID_CONVERT	regrid.convert
TC_RMW_REGRID_CENSOR_THRESH	regrid.censor_thresh
TC_RMW_REGRID_CENSOR_VAL	regrid.censor_val

${METPLUS_N_RANGE}

METplus Config(s)	MET Config File
TC_RMW_N_RANGE	n_range

${METPLUS_N_AZIMUTH}

METplus Config(s)	MET Config File
TC_RMW_N_AZIMUTH	n_azimuth

${METPLUS_MAX_RANGE_KM}

METplus Config(s)	MET Config File
TC_RMW_MAX_RANGE_KM	max_range_km

${METPLUS_DELTA_RANGE_KM}

METplus Config(s)	MET Config File
TC_RMW_DELTA_RANGE_KM	delta_range_km

${METPLUS_RMW_SCALE}

METplus Config(s)	MET Config File
TC_RMW_SCALE	rmw_scale

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
TC_RMW_MET_CONFIG_OVERRIDES	n/a

6.33 TCStat

6.33.1 Description

Used to configure the MET tool tc_stat.

6.33.2 METplus Configuration

TC_STAT_LOOKIN_DIR
TC_STAT_OUTPUT_DIR
TC_STAT_OUTPUT_TEMPLATE
TC_STAT_CONFIG_FILE
TC_STAT_JOB_ARGS
TC_STAT_AMODEL
TC_STAT_BMODEL
TC_STAT_DESC
TC_STAT_STORM_ID
TC_STAT_BASIN
TC_STAT_CYCLONE
TC_STAT_STORM_NAME
TC_STAT_INIT_BEG
TC_STAT_INIT_INCLUDE
TC_STAT_INIT_EXCLUDE
TC_STAT_INIT_HOUR
TC_STAT_VALID_BEG
TC_STAT_VALID_END
TC_STAT_VALID_INCLUDE
TC_STAT_VALID_EXCLUDE
TC_STAT_VALID_HOUR
TC_STAT_LEAD_REQ
TC_STAT_INIT_MASK
TC_STAT_VALID_MASK
TC_STAT_VALID_HOUR
TC_STAT_LEAD
TC_STAT_TRACK_WATCH_WARN
TC_STAT_COLUMN_THRESH_NAME
TC_STAT_COLUMN_THRESH_VAL
TC_STAT_COLUMN_STR_NAME
TC_STAT_COLUMN_STR_VAL
TC_STAT_INIT_THRESH_NAME
TC_STAT_INIT_THRESH_VAL

TC_STAT_INIT_STR_NAME
TC_STAT_INIT_STR_VAL
TC_STAT_WATER_ONLY
TC_STAT_LANDFALL
TC_STAT_LANDFALL_BEG
TC_STAT_LANDFALL_END
TC_STAT_MATCH_POINTS
TC_STAT_SKIP_IF_OUTPUT_EXISTS
TC_STAT_MET_CONFIG_OVERRIDES
TC_STAT_COLUMN_STR_EXC_NAME
TC_STAT_COLUMN_STR_EXC_VAL
TC_STAT_INIT_STR_EXC_NAME
TC_STAT_INIT_STR_EXC_VAL
TC_STAT_DIAG_THRESH_NAME
TC_STAT_DIAG_THRESH_VAL
TC_STAT_INIT_DIAG_THRESH_NAME
TC_STAT_INIT_DIAG_THRESH_VAL
TC_STAT_LINE_TYPE
TC_STAT_EVENT_EQUAL
TC_STAT_EVENT_EQUAL_LEAD
TC_STAT_OUT_INIT_MASK
TC_STAT_OUT_VALID_MASK

Warning: DEPRECATED:

TC_STAT_INPUT_DIR
TC_STAT_RUN_VIA
TC_STAT_CMD_LINE_JOB
TC_STAT_JOBS_LIST

6.33.3 MET Configuration

Below is the wrapped MET configuration file used for this wrapper. Environment variables are used to control entries in this configuration file. The default value for each environment variable is obtained from (except where noted below):

MET_INSTALL_DIR/share/met/config/TCStatConfig_default

Below the file contents are descriptions of each environment variable referenced in this file and the corresponding METplus configuration item used to set the value of the environment variable. For detailed examples showing how METplus sets the values of these environment variables, see *How METplus controls MET config file settings* (page 74).

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCStat configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to filter the TC-STAT data down to the
// desired subset of lines over which statistics are to be computed.  Only
// those lines which meet ALL of the criteria specified will be retained.
//
// The settings that are common to all jobs may be specified once at the top
// level.  If no selection is listed for a parameter, that parameter will not
// be used for filtering.  If multiple selections are listed for a parameter,
// the analyses will be performed on their union.
//

//
// Stratify by the AMODEL or BMODEL columns.
//
${METPLUS_AMODEL}
${METPLUS_BMODEL}

//
// Stratify by the DESC column.
//
${METPLUS_DESC}

//
// Stratify by the STORM_ID column.
//
${METPLUS_STORM_ID}

//
// Stratify by the BASIN column.
```

(continues on next page)

(continued from previous page)

```
// May add using the "-basin" job command option.
//
${METPLUS_BASIN}

//
// Stratify by the CYCLONE column.
// May add using the "-cyclone" job command option.
//
${METPLUS_CYCLONE}

//
// Stratify by the STORM_NAME column.
// May add using the "-storm_name" job command option.
//
${METPLUS_STORM_NAME}

//
// Stratify by the INIT times.
// Model initialization time windows to include or exclude
// May modify using the "-init_beg", "-init_end", "-init_inc",
// and "-init_exc" job command options.
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
${METPLUS_INIT_INC}
${METPLUS_INIT_EXC}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INC}
${METPLUS_VALID_EXC}

//
// Stratify by the initialization and valid hours and lead time.
//
${METPLUS_INIT_HOUR}
${METPLUS_VALID_HOUR}
${METPLUS_LEAD}

//
// Select tracks which contain all required lead times.
//
```

(continues on next page)

(continued from previous page)

```
${METPLUS_LEAD_REQ}

//
// Stratify by the INIT_MASK and VALID_MASK columns.
//
${METPLUS_INIT_MASK}
${METPLUS_VALID_MASK}


//
// Stratify by the LINE_TYPE column.
//
//line_type =
${METPLUS_LINE_TYPE}


//
// Stratify by checking the watch/warning status for each track point
// common to both the ADECK and BDECK tracks.  If the watch/warning status
// of any of the track points appears in the list, retain the entire track.
//
${METPLUS_TRACK_WATCH_WARN}

//
// Stratify by applying thresholds to numeric data columns.
//
${METPLUS_COLUMN_THRESH_NAME}
${METPLUS_COLUMN_THRESH_VAL}

//
// Stratify by performing string matching on non-numeric data columns.
//
${METPLUS_COLUMN_STR_NAME}
${METPLUS_COLUMN_STR_VAL}

//
// Stratify by excluding strings in non-numeric data columns.
//
//column_str_exc_name =
${METPLUS_COLUMN_STR_EXC_NAME}

//column_str_exc_val =
${METPLUS_COLUMN_STR_EXC_VAL}

//
```

(continues on next page)

(continued from previous page)

```
// Similar to the column_thresh options above
//
${METPLUS_INIT_THRESH_NAME}
${METPLUS_INIT_THRESH_VAL}

//
// Similar to the column_str options above
//
${METPLUS_INIT_STR_NAME}
${METPLUS_INIT_STR_VAL}

//
// Similar to the column_str_exc options above
//
//init_str_exc_name =
${METPLUS_INIT_STR_EXC_NAME}

//init_str_exc_val =
${METPLUS_INIT_STR_EXC_VAL}

//diag_thresh_name =
${METPLUS_DIAG_THRESH_NAME}

//diag_thresh_val =
${METPLUS_DIAG_THRESH_VAL}

//init_diag_thresh_name =
${METPLUS_INIT_DIAG_THRESH_NAME}

//init_diag_thresh_val =
${METPLUS_INIT_DIAG_THRESH_VAL}


//
// Stratify by the ADECK and BDECK distances to land.
//
${METPLUS_WATER_ONLY}

//
// Specify whether only those track points occurring near landfall should be
// retained, and define the landfall retention window in HH[MMSS] format
// around the landfall time.
//
${METPLUS_LANDFALL}
${METPLUS_LANDFALL_BEG}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_LANDFALL_END}

//
// Specify whether only those track points common to both the ADECK and BDECK
// tracks should be retained.  May modify using the "-match_points" job command
// option.
//
${METPLUS_MATCH_POINTS}

//event_equal =
${METPLUS_EVENT_EQUAL}

//event_equal_lead =
${METPLUS_EVENT_EQUAL_LEAD}

//out_init_mask =
${METPLUS_OUT_INIT_MASK}

//out_valid_mask =
${METPLUS_OUT_VALID_MASK}


//
// Array of TCStat analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

${METPLUS_AMODEL}

METplus Config(s)	MET Config File
TC_STAT_AMODEL	amodel

${METPLUS_BMODEL}

METplus Config(s)	MET Config File
TC_STAT_BMODEL	bmodel

${METPLUS_DESC}

METplus Config(s)	MET Config File
TC_STAT_DESC	desc

${METPLUS_STORM_ID}

METplus Config(s)	MET Config File
TC_STAT_STORM_ID	storm_id

${METPLUS_BASIN}

METplus Config(s)	MET Config File
TC_STAT_BASIN	basin

${METPLUS_CYCLONE}

METplus Config(s)	MET Config File
TC_STAT_CYCLONE	cyclone

${METPLUS_STORM_NAME}

METplus Config(s)	MET Config File
TC_STAT_STORM_NAME	storm_name

${METPLUS_INIT_BEG}

METplus Config(s)	MET Config File
TC_STAT_INIT_BEG	init_beg

${METPLUS_INIT_END}

METplus Config(s)	MET Config File
TC_STAT_INIT_END	init_end

${METPLUS_INIT_INCLUDE}

METplus Config(s)	MET Config File
TC_STAT_INIT_INCLUDE	init_inc

${METPLUS_INIT_EXCLUDE}

METplus Config(s)	MET Config File
TC_STAT_INIT_EXCLUDE	init_exc

${METPLUS_VALID_BEG}

METplus Config(s)	MET Config File
TC_STAT_VALID_BEG	valid_beg

${METPLUS_VALID_END}

METplus Config(s)	MET Config File
TC_STAT_VALID_END	valid_end

${METPLUS_VALID_INCLUDE}

METplus Config(s)	MET Config File
TC_STAT_VALID_INCLUDE	valid_inc

${METPLUS_VALID_EXCLUDE}

METplus Config(s)	MET Config File
TC_STAT_VALID_EXCLUDE	valid_exc

${METPLUS_INIT_HOUR}

METplus Config(s)	MET Config File
TC_STAT_INIT_HOUR	init_hour

${METPLUS_VALID_HOUR}

METplus Config(s)	MET Config File
TC_STAT_VALID_HOUR	valid_hour

${METPLUS_LEAD}

METplus Config(s)	MET Config File
TC_STAT_LEAD	lead

${METPLUS_LEAD_REQ}

METplus Config(s)	MET Config File
TC_STAT_LEAD_REQ	lead_req

${METPLUS_INIT_MASK}

METplus Config(s)	MET Config File
TC_STAT_INIT_MASK	init_mask

${METPLUS_VALID_MASK}

METplus Config(s)	MET Config File
TC_STAT_VALID_MASK	valid_mask

${METPLUS_LINE_TYPE}

METplus Config(s)	MET Config File
TC_STAT_LINE_TYPE	line_type

${METPLUS_TRACK_WATCH_WARN}

METplus Config(s)	MET Config File
TC_STAT_TRACK_WATCH_WARN	track_watch_warn

${METPLUS_COLUMN_THRESH_NAME}

METplus Config(s)	MET Config File
TC_STAT_COLUMN_THRESH_NAME	column_thresh_name

${METPLUS_COLUMN_THRESH_VAL}

METplus Config(s)	MET Config File
TC_STAT_COLUMN_THRESH_VAL	column_thresh_val

${METPLUS_COLUMN_STR_NAME}

METplus Config(s)	MET Config File
TC_STAT_COLUMN_STR_NAME	column_str_name

${METPLUS_COLUMN_STR_VAL}

METplus Config(s)	MET Config File
TC_STAT_COLUMN_STR_VAL	column_str_val

${METPLUS_COLUMN_STR_EXC_NAME}

METplus Config(s)	MET Config File
TC_STAT_COLUMN_STR_EXC_NAME	column_str_exc_name

${METPLUS_COLUMN_STR_EXC_VAL}

METplus Config(s)	MET Config File
TC_STAT_COLUMN_STR_EXC_VAL	column_str_exc_val

${METPLUS_INIT_THRESH_NAME}

METplus Config(s)	MET Config File
TC_STAT_INIT_THRESH_NAME	init_thresh_name

${METPLUS_INIT_THRESH_VAL}

METplus Config(s)	MET Config File
TC_STAT_INIT_THRESH_VAL	init_thresh_val

${METPLUS_INIT_STR_NAME}

METplus Config(s)	MET Config File
TC_STAT_INIT_STR_NAME	init_str_name

${METPLUS_INIT_STR_VAL}

METplus Config(s)	MET Config File
TC_STAT_INIT_STR_VAL	init_str_val

${METPLUS_INIT_STR_EXC_NAME}

METplus Config(s)	MET Config File
TC_STAT_INIT_STR_EXC_NAME	init_str_exc_name

${METPLUS_INIT_STR_EXC_VAL}

METplus Config(s)	MET Config File
TC_STAT_INIT_STR_EXC_VAL	init_str_exc_val

${METPLUS_DIAG_THRESH_NAME}

METplus Config(s)	MET Config File
TC_STAT_DIAG_THRESH_NAME	diag_thresh_name

${METPLUS_DIAG_THRESH_VAL}

METplus Config(s)	MET Config File
TC_STAT_DIAG_THRESH_VAL	diag_thresh_val

${METPLUS_INIT_DIAG_THRESH_NAME}

METplus Config(s)	MET Config File
TC_STAT_INIT_DIAG_THRESH_NAME	init_diag_thresh_name

${METPLUS_INIT_DIAG_THRESH_VAL}

METplus Config(s)	MET Config File
TC_STAT_INIT_DIAG_THRESH_VAL	init_diag_thresh_val

${METPLUS_WATER_ONLY}

METplus Config(s)	MET Config File
TC_STAT_WATER_ONLY	water_only

${METPLUS_LANDFALL}

METplus Config(s)	MET Config File
TC_STAT_LANDFALL	landfall

${METPLUS_LANDFALL_BEG}

METplus Config(s)	MET Config File
TC_STAT_LANDFALL_BEG	landfall_beg

${METPLUS_LANDFALL_END}

METplus Config(s)	MET Config File
TC_STAT_LANDFALL_END	landfall_end

${METPLUS_MATCH_POINTS}

METplus Config(s)	MET Config File
TC_STAT_MATCH_POINTS	match_points

${METPLUS_JOBS}

METplus Config(s)	MET Config File
TC_STAT_JOBS_LIST	jobs

${METPLUS_MET_CONFIG_OVERRIDES}

METplus Config(s)	MET Config File
TC_STAT_MET_CONFIG_OVERRIDES	n/a

${METPLUS_EVENT_EQUAL}

METplus Config(s)	MET Config File
TC_STAT_EVENT_EQUAL	event_equal

${METPLUS_EVENT_EQUAL_LEAD}

METplus Config(s)	MET Config File
TC_STAT_EVENT_EQUAL_LEAD	event_equal_lead

${METPLUS_OUT_INIT_MASK}

METplus Config(s)	MET Config File
TC_STAT_OUT_INIT_MASK	out_init_mask

${METPLUS_OUT_VALID_MASK}

METplus Config(s)	MET Config File
TC_STAT_OUT_VALID_MASK	out_valid_mask

6.34 UserScript

6.34.1 Description

Used to generate user-defined commands to run in the process list. Commands can be run once, run once for each runtime (init/valid/lead combination) or once for init, valid, or lead only. The command to run is specified with the *USER_SCRIPT_COMMAND* variable. The command should include a script or executable and any desired arguments. The variable support filename template substitution to send information like the current initialization or forecast lead time. See *Runtime Frequency* (page 70) for more information on how the value of *USER_SCRIPT_RUNTIME_FREQ* can control how the commands are called. Optionally, file paths can be defined with filename templates to generate a file list text file that contains all existing file paths that correspond to the appropriate runtime frequency for the current run time. The path to the file list text files are set as environment variables that can be referenced inside the user-defined script to obtain a list of the files that should be processed. See *USER_SCRIPT_INPUT_TEMPLATE* for more information.

Note: This wrapper may be disabled upon installation to prevent security risks.

6.34.2 METplus Configuration

USER_SCRIPT_RUNTIME_FREQ
USER_SCRIPT_COMMAND
USER_SCRIPT_CUSTOM_LOOP_LIST
USER_SCRIPT_SKIP_TIMES
USER_SCRIPT_INPUT_DIR
USER_SCRIPT_INPUT_TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS

Chapter 7

METplus Use Cases

The METplus Use Cases provide a low-level workflow which includes setting paths to data, dates to process, the order of processing, and configuration options. The METplus Wrappers pass data and configuration options to the MET tools.

7.1 MET tools

7.1.1 ASCII2NC

7.1.2 Cyclone Plotter

7.1.3 EnsembleStat

7.1.4 Example

7.1.5 ExtractTiles

7.1.6 GFDLTracker

7.1.7 GempakToCF

7.1.8 GenEnsProd

7.1.9 GenVxMask

7.1.10 GridDiag

7.1.11 GridStat

7.1.12 IODA2NC

7.1.13 METdbLoad

7.1.14 MODE

7.1.15 MTD

7.1.16 PB2NC

7.1.17 PCPCombine

7.1.18 PlotDataPlane

7.1.19 PlotPointObs

7.1.20 Point2Grid

7.1.21 PointStat

7.1.22 PyEmbedIngest

7.1.23 RegridDataPlane

7.1.24 SeriesAnalysis

7.1.25 StatAnalysis

Scientific Objective

None. Simply converting file formats so point observations can be read by the MET tools.

Datasets

Observations: Precipitation accumulation observations in ASCII text files

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 316) section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus ASCII2NC wrapper to generate a command to run the MET tool ASCII2NC if all required files are found.

METplus Workflow

ASCII2NC is the only tool called in this example. It processes the following run time:

Valid: 2010-01-01_12Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/ASCII2NC/ASCII2NC.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/ASCII2NC/ASCII2NC.html
```

(continues on next page)

(continued from previous page)

```
# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = ASCII2NC


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2010010112
VALID_END = 2010010112
VALID_INCREMENT = 1M

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

ASCII2NC_INPUT_DIR =
ASCII2NC_INPUT_TEMPLATE = {INPUT_BASE}/met_test/data/sample_obs/ascii/precip24_{valid?fmt=%Y
↪%m%d%H}.ascii

ASCII2NC_OUTPUT_DIR =
ASCII2NC_OUTPUT_TEMPLATE = {OUTPUT_BASE}/ascii2nc/precip24_{valid?fmt=%Y%m%d%H}.nc
```

(continues on next page)

(continued from previous page)

```
ASCII2NC_SKIP_IF_OUTPUT_EXISTS = False

ASCII2NC_FILE_WINDOW_BEGIN = 0
ASCII2NC_FILE_WINDOW_END = 0

###
# ASCII2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ascii2nc
###

#LOG_ASCII2NC_VERBOSITY = 1

ASCII2NC_CONFIG_FILE = {PARM_BASE}/met_config/Ascii2NcConfig_wrapped

ASCII2NC_INPUT_FORMAT =

ASCII2NC_MASK_GRID =
ASCII2NC_MASK_POLY =
ASCII2NC_MASK_SID =

ASCII2NC_TIME_SUMMARY_FLAG = False
ASCII2NC_TIME_SUMMARY_RAW_DATA = False
ASCII2NC_TIME_SUMMARY_BEG = 000000
ASCII2NC_TIME_SUMMARY_END = 235959
ASCII2NC_TIME_SUMMARY_STEP = 300
ASCII2NC_TIME_SUMMARY_WIDTH = 600
ASCII2NC_TIME_SUMMARY_GRIB_CODES = 11, 204, 211
ASCII2NC_TIME_SUMMARY_VAR_NAMES =
ASCII2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80
ASCII2NC_TIME_SUMMARY_VALID_FREQ = 0
ASCII2NC_TIME_SUMMARY_VALID_THRESH = 0.0
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *ASCII2NC MET Configuration* (page 98) section of the User's Guide for more information on

the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default ascii2nc configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to summarize the ASCII data read in
//

//
// Time periods for the summarization
// obs_var (string array) is added and works like grib_code (int array)
// when the obs name is given instead of grib_code
//
${METPLUS_TIME_SUMMARY_DICT}

//
// Mapping of input little_r report types to output message types
//
message_type_map = [
   { key = "FM-12 SYNOP";  val = "ADPSFC"; },
   { key = "FM-13 SHIP";   val = "SFCSHP"; },
   { key = "FM-15 METAR";  val = "ADPSFC"; },
   { key = "FM-18 BUOY";   val = "SFCSHP"; },
   { key = "FM-281 QSCAT"; val = "ASCATW"; },
   { key = "FM-32 PILOT";  val = "ADPUPA"; },
   { key = "FM-35 TEMP";   val = "ADPUPA"; },
   { key = "FM-88 SATOB";  val = "SATWND"; },
   { key = "FM-97 ACARS";  val = "AIRCFT"; }
];

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in ASCII2NC.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/ASCII2NC/ASCII2NC.
↪conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in ASCII2NC.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/ASCII2NC/ASCII2NC.
↪conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in ascii2nc (relative to **OUTPUT_BASE**) and will contain the following file:

- precip24_2010010112.nc

Keywords

Note:

- ASCII2NCToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-ASCII2NC.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.1.2 ASCII2NC: Using Python Embedding

met_tool_wrapper/ASCII2NC/ASCII2NC_python _embedding.conf

Scientific Objective

Simply converting file formats so point observations can be read by the MET tools through the use of a Python script

Datasets

Observations: Precipitation accumulation observations in ASCII text files

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 320) section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus ASCII2NC wrapper to generate a command to run the MET tool ASCII2NC.

METplus Workflow

ASCII2NC is the only tool called in this example. It has one run time, but the time is not relevant because the files processed do not have any time information in the names.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/ASCII2NC/ASCII2NC_python_embedding.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/ASCII2NC/ASCII2NC_
↪python_embedding.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = ASCII2NC


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
```

(continues on next page)

(continued from previous page)

```
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2010010112
VALID_END = 2010010112
VALID_INCREMENT = 1M

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

ASCII2NC_INPUT_DIR =
ASCII2NC_INPUT_TEMPLATE = "{MET_INSTALL_DIR}/share/met/python/examples/read_ascii_point.py
↪{INPUT_BASE}/met_test/data/sample_obs/ascii/sample_ascii_obs.txt"

ASCII2NC_OUTPUT_DIR =
ASCII2NC_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/ASCII2NC/ascii2nc_python.nc

ASCII2NC_SKIP_IF_OUTPUT_EXISTS = False

ASCII2NC_FILE_WINDOW_BEGIN = 0
ASCII2NC_FILE_WINDOW_END = 0


###
# ASCII2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ascii2nc
###

#LOG_ASCII2NC_VERBOSITY = 1
#ASCII2NC_CONFIG_FILE =

ASCII2NC_WINDOW_BEGIN = 0
ASCII2NC_WINDOW_END = 0

ASCII2NC_INPUT_FORMAT = python

ASCII2NC_MASK_GRID =
ASCII2NC_MASK_POLY =
ASCII2NC_MASK_SID =

ASCII2NC_TIME_SUMMARY_FLAG = False
```

(continues on next page)

(continued from previous page)

```
ASCII2NC_TIME_SUMMARY_RAW_DATA = False
ASCII2NC_TIME_SUMMARY_BEG = 000000
ASCII2NC_TIME_SUMMARY_END = 235959
ASCII2NC_TIME_SUMMARY_STEP = 300
ASCII2NC_TIME_SUMMARY_WIDTH = 600
ASCII2NC_TIME_SUMMARY_GRIB_CODES = 11, 204, 211
ASCII2NC_TIME_SUMMARY_VAR_NAMES =
ASCII2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80
ASCII2NC_TIME_SUMMARY_VALID_FREQ = 0
ASCII2NC_TIME_SUMMARY_VALID_THRESH = 0.0
```

MET Configuration

None. No MET configuration file for ASCII2NC is used in this case.

Python Embedding

This use case calls a Python script to read the input data. The Python script is stored in the MET repository: /path/to/MET/installation/share/met/python/read_ascii_point.py

read_ascii_point.py

Running METplus

This use case can be run two ways:

1) Passing in ASCII2NC_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/ASCII2NC/ASCII2NC_
↪python_embedding.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in ASCII2NC_python_embedding.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/ASCII2NC/ASCII2NC_
↪python_embedding.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/ASCII2NC (relative to **OUTPUT_BASE**) and will contain the following file:

- ascii2nc_python.nc

Keywords

Note:

- ASCII2NCToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-ASCII2NC.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.2 Cyclone Plotter

7.1.32.2.1 CyclonePlotter: Basic Use Case

met_tool_wrapper/CyclonePlotter/CyclonePlotter.conf

Scientific Objective

Provide visualization of cyclone tracks on a global map (PlateCaree projection)

Datasets

No datasets are required for running this use case. Only output from running the MET Tool tc-pairs or the METplus tc pairs wrapper is required.

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- cartopy
- matplotlib

METplus Components

This use case does not utilize any MET tools

METplus Workflow

CyclonePlotter is the only tool called in this example. It processes the following run times:

Init: 2015-03-01_12Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/CyclonePlotter/CyclonePlotter.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/CyclonePlotter/
↪CyclonePlotter.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide
```

(continues on next page)

(continued from previous page)

```
###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = CyclonePlotter


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

CYCLONE_PLOTTER_INPUT_DIR = {INPUT_BASE}/met_test/tc_pairs

CYCLONE_PLOTTER_OUTPUT_DIR = {OUTPUT_BASE}/cyclone


###
# CyclonePlotter Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#cycloneplotter
###

CYCLONE_PLOTTER_INIT_DATE = 20150301

CYCLONE_PLOTTER_INIT_HR = 12
CYCLONE_PLOTTER_MODEL = GFSO
CYCLONE_PLOTTER_PLOT_TITLE = Model Forecast Storm Tracks

CYCLONE_PLOTTER_GLOBAL_PLOT = no

CYCLONE_PLOTTER_WEST_LON = -180
CYCLONE_PLOTTER_EAST_LON = 179
CYCLONE_PLOTTER_SOUTH_LAT = 0
CYCLONE_PLOTTER_NORTH_LAT = 90


CYCLONE_PLOTTER_CIRCLE_MARKER_SIZE = 2
CYCLONE_PLOTTER_CROSS_MARKER_SIZE = 11

CYCLONE_PLOTTER_ANNOTATION_FONT_SIZE = 3

CYCLONE_PLOTTER_RESOLUTION_DPI = 400
```

(continues on next page)

(continued from previous page)

```
CYCLONE_PLOTTER_GENERATE_TRACK_ASCII = yes

CYCLONE_PLOTTER_ADD_WATERMARK = False
```

MET Configuration

No MET configuration is needed to run the cyclone plotter wrapper.

Running METplus

This use case can be run as follows:

1) Passing in CyclonePlotter.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/CyclonePlotter/
↪CyclonePlotter.conf \
                 -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in CyclonePlotter.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/CyclonePlotter/
↪CyclonePlotter.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will generate output to both the screen and to the logfile:

INFO: METplus has successfully finished running.

Additionally, two output files are created. Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. TCPairs output for this use case will be found in cyclone/201503 (relative to **OUTPUT_BASE**) and will contain files with the following format:

- 20150301.txt
- 20150301.png

Keywords

Note:

- CyclonePlotterUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.3 EnsembleStat

7.1.32.3.1 EnsembleStat: Basic Use Case

met_tool_wrapper/EnsembleStat/EnsembleStat.conf

Scientific Objective

To provide useful statistical information on the relationship between observation data (in both grid and point formats) to an ensemble forecast. These values can be used to help correct ensemble member deviations from observed values.

Datasets

Forecast: WRF ARW 24 hour precipitation accumulation

. . . met_test/data/sample_fcst/2009123112/
arw-fer-gep1/d01_2009123112_02400.grib
arw-fer-gep5/d01_2009123112_02400.grib
arw-sch-gep2/d01_2009123112_02400.grib
arw-sch-gep6/d01_2009123112_02400.grib

arw-tom-gep3/d01_2009123112_02400.grib
arw-tom-gep7/d01_2009123112_02400.grib

Gridded Observation: ST4 24 hour precipitation accumulation

met_test/data/sample_obs/ST4/sample_obs/ST4/ST4.2010010112.24h

Point Observation:

met_test/out/ascii2nc/precip24_2010010112.nc

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 337) section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus EnsembleStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool EnsembleStat if all required files are found.

METplus Workflow

EnsembleStat is the only tool called in this example. It processes the following run times:

Init: 2009-12-31_12Z
Forecast lead: 24 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/EnsembleStat/EnsembleStat.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/EnsembleStat/
↪EnsembleStat.html

```

(continues on next page)

(continued from previous page)

```
# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = EnsembleStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2009123112
INIT_END=2009123112
INIT_INCREMENT=3600

LEAD_SEQ = 24H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_ENSEMBLE_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/arw-???-gep?/d01_{init?fmt=%Y%m%d%H}_
↪0{lead?fmt=%HH}00.grib

#ENSEMBLE_STAT_CTRL_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
#ENSEMBLE_STAT_CTRL_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/arw-fer-gep1/d01_{init?fmt=%Y%m%d%H}
↪_0{lead?fmt=%HH}00.grib
```

(continues on next page)

(continued from previous page)

```
OBS_ENSEMBLE_STAT_POINT_INPUT_DIR = {INPUT_BASE}/met_test/out/ascii2nc
OBS_ENSEMBLE_STAT_POINT_INPUT_TEMPLATE = precip24_{valid?fmt=%Y%m%d%H}.nc

OBS_ENSEMBLE_STAT_GRID_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_obs/ST4
OBS_ENSEMBLE_STAT_GRID_INPUT_TEMPLATE = ST4.{valid?fmt=%Y%m%d%H}.24h

#ENSEMBLE_STAT_ENS_MEAN_INPUT_DIR =
#ENSEMBLE_STAT_ENS_MEAN_INPUT_TEMPLATE =

ENSEMBLE_STAT_CLIMO_MEAN_INPUT_DIR =
ENSEMBLE_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

ENSEMBLE_STAT_CLIMO_STDEV_INPUT_DIR =
ENSEMBLE_STAT_CLIMO_STDEV_INPUT_TEMPLATE =


ENSEMBLE_STAT_OUTPUT_DIR = {OUTPUT_BASE}/ensemble
ENSEMBLE_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H%M}/ensemble_stat


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = WRF
OBTYPE = MC_PCP


FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A24
FCST_VAR1_OPTIONS = ens_ssvar_bin_size = 0.1; ens_phist_bin_size = 0.05;


OBS_VAR1_NAME = {FCST_VAR1_NAME}
OBS_VAR1_LEVELS = {FCST_VAR1_LEVELS}
OBS_VAR1_OPTIONS = {FCST_VAR1_OPTIONS}


###
# EnsembleStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ensemblestat
###
```

(continues on next page)

(continued from previous page)

```
#LOG_ENSEMBLE_STAT_VERBOSITY = 2

ENSEMBLE_STAT_N_MEMBERS = 6

ENSEMBLE_STAT_CONFIG_FILE = {PARM_BASE}/met_config/EnsembleStatConfig_wrapped

ENSEMBLE_STAT_DESC = NA

ENSEMBLE_STAT_OBS_WINDOW_BEGIN = -5400
ENSEMBLE_STAT_OBS_WINDOW_END = 5400

ENSEMBLE_STAT_ENS_THRESH = 1.0

ENSEMBLE_STAT_VLD_THRESH = 1.0

ENSEMBLE_STAT_OUTPUT_PREFIX =

#ENSEMBLE_STAT_MET_OBS_ERR_TABLE =

ENSEMBLE_STAT_REGRID_TO_GRID = NONE
ENSEMBLE_STAT_REGRID_METHOD = NEAREST
ENSEMBLE_STAT_REGRID_WIDTH = 1
ENSEMBLE_STAT_REGRID_VLD_THRESH = 0.5
ENSEMBLE_STAT_REGRID_SHAPE = SQUARE
#ENSEMBLE_STAT_REGRID_CONVERT =
#ENSEMBLE_STAT_REGRID_CENSOR_THRESH =
#ENSEMBLE_STAT_REGRID_CENSOR_VAL =

ENSEMBLE_STAT_CENSOR_THRESH =
ENSEMBLE_STAT_CENSOR_VAL =

#ENSEMBLE_STAT_PROB_CAT_THRESH =
#ENSEMBLE_STAT_PROB_PCT_THRESH = ==0.25
#ENSEMBLE_STAT_ECLV_POINTS = 0.05

ENSEMBLE_STAT_MESSAGE_TYPE = ADPSFC
#ENSEMBLE_STAT_OBS_THRESH =
ENSEMBLE_STAT_DUPLICATE_FLAG = NONE
ENSEMBLE_STAT_SKIP_CONST = False

ENSEMBLE_STAT_OBS_ERROR_FLAG = FALSE

ENSEMBLE_STAT_ENS_SSVAR_BIN_SIZE = 1.0
ENSEMBLE_STAT_ENS_PHIST_BIN_SIZE = 0.05
```

(continues on next page)

(continued from previous page)

```
#ENSEMBLE_STAT_CLIMO_MEAN_FILE_NAME =
#ENSEMBLE_STAT_CLIMO_MEAN_FIELD =
#ENSEMBLE_STAT_CLIMO_MEAN_REGRID_METHOD =
#ENSEMBLE_STAT_CLIMO_MEAN_REGRID_WIDTH =
#ENSEMBLE_STAT_CLIMO_MEAN_REGRID_VLD_THRESH =
#ENSEMBLE_STAT_CLIMO_MEAN_REGRID_SHAPE =
#ENSEMBLE_STAT_CLIMO_MEAN_TIME_INTERP_METHOD =
#ENSEMBLE_STAT_CLIMO_MEAN_MATCH_MONTH =
#ENSEMBLE_STAT_CLIMO_MEAN_DAY_INTERVAL = 31
#ENSEMBLE_STAT_CLIMO_MEAN_HOUR_INTERVAL = 6

#ENSEMBLE_STAT_CLIMO_STDEV_FILE_NAME =
#ENSEMBLE_STAT_CLIMO_STDEV_FIELD =
#ENSEMBLE_STAT_CLIMO_STDEV_REGRID_METHOD =
#ENSEMBLE_STAT_CLIMO_STDEV_REGRID_WIDTH =
#ENSEMBLE_STAT_CLIMO_STDEV_REGRID_VLD_THRESH =
#ENSEMBLE_STAT_CLIMO_STDEV_REGRID_SHAPE =
#ENSEMBLE_STAT_CLIMO_STDEV_TIME_INTERP_METHOD =
#ENSEMBLE_STAT_CLIMO_STDEV_MATCH_MONTH =
#ENSEMBLE_STAT_CLIMO_STDEV_DAY_INTERVAL = 31
#ENSEMBLE_STAT_CLIMO_STDEV_HOUR_INTERVAL = 6

ENSEMBLE_STAT_CLIMO_CDF_BINS = 1
ENSEMBLE_STAT_CLIMO_CDF_CENTER_BINS = False
ENSEMBLE_STAT_CLIMO_CDF_WRITE_BINS = True
#ENSEMBLE_STAT_CLIMO_CDF_DIRECT_PROB =

ENSEMBLE_STAT_MASK_GRID = FULL
ENSEMBLE_STAT_MASK_POLY =
    MET_BASE/poly/HMT_masks/huc4_1605_poly.nc,
    MET_BASE/poly/HMT_masks/huc4_1803_poly.nc,
    MET_BASE/poly/HMT_masks/huc4_1804_poly.nc,
    MET_BASE/poly/HMT_masks/huc4_1805_poly.nc,
    MET_BASE/poly/HMT_masks/huc4_1806_poly.nc

ENSEMBLE_STAT_CI_ALPHA = 0.05

ENSEMBLE_STAT_INTERP_FIELD = BOTH
ENSEMBLE_STAT_INTERP_VLD_THRESH = 1.0
ENSEMBLE_STAT_INTERP_SHAPE = SQUARE
ENSEMBLE_STAT_INTERP_METHOD = NEAREST
ENSEMBLE_STAT_INTERP_WIDTH = 1

ENSEMBLE_STAT_OUTPUT_FLAG_ECNT = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_RPS = NONE
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_OUTPUT_FLAG_RHIST = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_PHIST = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_ORANK = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_SSVAR = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_RELP = BOTH
#ENSEMBLE_STAT_OUTPUT_FLAG_PCT = BOTH
#ENSEMBLE_STAT_OUTPUT_FLAG_PSTD = BOTH
#ENSEMBLE_STAT_OUTPUT_FLAG_PJC = BOTH
#ENSEMBLE_STAT_OUTPUT_FLAG_PRC = BOTH
#ENSEMBLE_STAT_OUTPUT_FLAG_ECLV = BOTH

ENSEMBLE_STAT_NC_ORANK_FLAG_LATLON = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_MEAN = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_RAW = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_RANK = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_PIT = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_VLD_COUNT = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_WEIGHT = FALSE

#ENSEMBLE_STAT_OBS_QUALITY_INC =
#ENSEMBLE_STAT_OBS_QUALITY_EXC =

#ENSEMBLE_STAT_ENS_MEMBER_IDS =
#ENSEMBLE_STAT_CONTROL_ID =

#ENSEMBLE_STAT_GRID_WEIGHT_FLAG =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *EnsembleStat MET Configuration* (page 105) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Ensemble-Stat configuration file.
```

(continues on next page)

(continued from previous page)

```
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
${METPLUS_DESC}

//
// Output observation type to be written
//
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
${METPLUS_CENSOR_THRESH}
${METPLUS_CENSOR_VAL}
cat_thresh    = [];
nc_var_str    = "";

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//prob_cat_thresh =
${METPLUS_PROB_CAT_THRESH}

//prob_pct_thresh =
${METPLUS_PROB_PCT_THRESH}

//eclv_points =
${METPLUS_ECLV_POINTS}


////////////////////////////////////////////////////////////////////////////////

//
// Forecast and observation fields to be verified
//

fcst = {

   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_ENS_THRESH}
   ${METPLUS_VLD_THRESH}
   ${METPLUS_FCST_FIELD}
}

obs = {

   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//

${METPLUS_MESSAGE_TYPE}
sid_exc        = [];
//obs_thresh     = [ NA ];
${METPLUS_OBS_THRESH}
```

(continues on next page)

(continued from previous page)

```
//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

${METPLUS_DUPLICATE_FLAG}
obs_summary    = NONE;
obs_perc_value = 50;
${METPLUS_SKIP_CONST}

//
// Observation error options
// Set dist_type to NONE to use the observation error table instead
// May be set separately in each "obs.field" entry
//
obs_error = {
   ${METPLUS_OBS_ERROR_FLAG}
   dist_type        = NONE;
   dist_parm        = [];
   inst_bias_scale  = 1.0;
   inst_bias_offset = 0.0;
   min              = NA;      // Valid range of data
   max              = NA;
}

//
// Mapping of message type group name to comma-separated list of values.
//
message_type_group_map = [
   { key = "SURFACE"; val = "ADPSFC,SFCSHP,MSONET";               },
   { key = "ANYAIR";  val = "AIRCAR,AIRCFT";                      },
   { key = "ANYSFC";  val = "ADPSFC,SFCSHP,ADPUPA,PROFLR,MSONET"; },
   { key = "ONLYSF";  val = "ADPSFC,SFCSHP";                      }
];

//
// Ensemble bin sizes
// May be set separately in each "obs.field" entry
//
${METPLUS_ENS_SSVAR_BIN_SIZE}
${METPLUS_ENS_PHIST_BIN_SIZE}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}



//
// May be set separately in each "obs.field" entry
//
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
mask = {
   ${METPLUS_MASK_GRID}
   ${METPLUS_MASK_POLY}
   sid   = [];
   llpnt = [];
}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
${METPLUS_CI_ALPHA}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Interpolation methods
//
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gridded verification output types
// May be set separately in each "obs.field" entry
//
${METPLUS_NC_ORANK_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Random number generator
//
rng = {
   type = "mt19937";
   seed = "1";
}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

${METPLUS_OUTPUT_PREFIX}
//version          = "V9.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

It is recommended to run this use case by:

Passing in EnsembleStat.conf then a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/met_tool_wrapper/EnsembleStat/EnsembleStat.
↪conf /path/to/user_system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in ensemble/200912311200/ensemble_stat (relative to **OUTPUT_BASE**) and will contain the following files:

- ensemble_stat_20100101_120000V.stat
- ensemble_stat_20100101_120000V_ecnt.txt
- ensemble_stat_20100101_120000V_rhist.txt
- ensemble_stat_20100101_120000V_phist.txt
- ensemble_stat_20100101_120000V_orank.txt
- ensemble_stat_20100101_120000V_ssvar.txt
- ensemble_stat_20100101_120000V_relp.txt
- ensemble_stat_20100101_120000V_ens.nc

- ensemble_stat_20100101_120000V_orank.nc

Keywords

Note:

- EnsembleStatToolUseCase
- PythonEmbeddingFileUseCase
- EnsembleAppUseCase
- ProbabilityGenerationAppUseCase
- GRIBFileUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1997) to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-EnsembleStat.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.3.2 EnsembleStat: Using Python Embedding

met_tool_wrapper/EnsembleStat/EnsembleStat_python _embedding.conf

Scientific Objective

To provide useful statistical information on the relationship between observation data (in both grid and point formats) to an ensemble forecast. These values can be used to help correct ensemble member deviations from observed values.

Datasets

Forecast: Dummy text files found in the MET shared directory
Observation: Dummy text files found in the MET shared directory

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 348) section for more information.

METplus Components

This use case utilizes the METplus EnsembleStat wrapper to read in files using Python Embedding to demonstrate how to read in data this way.

METplus Workflow

EnsembleStat is the only tool called in this example. It processes a single run time with two ensemble members. The input data are simple text files with no timing information, so the list of ensembles simply duplicates the same file multiple times to demonstrate how data is read in via Python Embedding.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/EnsembleStat/EnsembleStat_python_embedding.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/EnsembleStat/
↪EnsembleStat_python_embedding.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = EnsembleStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###
```

(continues on next page)

(continued from previous page)

```
LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2009123112
INIT_END=2009123112
INIT_INCREMENT=3600

LEAD_SEQ = 24


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_ENSEMBLE_STAT_INPUT_DATATYPE = PYTHON_NUMPY

OBS_ENSEMBLE_STAT_INPUT_GRID_DATATYPE = PYTHON_NUMPY

FCST_ENSEMBLE_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/python
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE = fcst.txt, fcst.txt

OBS_ENSEMBLE_STAT_POINT_INPUT_DIR =
OBS_ENSEMBLE_STAT_POINT_INPUT_TEMPLATE =

OBS_ENSEMBLE_STAT_GRID_INPUT_DIR = {INPUT_BASE}/met_test/data/python
OBS_ENSEMBLE_STAT_GRID_INPUT_TEMPLATE = obs.txt

ENSEMBLE_STAT_CLIMO_MEAN_INPUT_DIR =
ENSEMBLE_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

ENSEMBLE_STAT_CLIMO_STDEV_INPUT_DIR =
ENSEMBLE_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

ENSEMBLE_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/EnsembleStat/ens_python_embedding
ENSEMBLE_STAT_OUTPUT_TEMPLATE =

OBS_FILE_WINDOW_BEGIN = 0
OBS_FILE_WINDOW_END = 0


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
```

(continues on next page)

(continued from previous page)

```
###

FCST_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py MET_PYTHON_
↪INPUT_ARG FCST
OBS_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py MET_PYTHON_
↪INPUT_ARG OBS

MODEL = FCST
OBTYPE = OBS


###
# EnsembleStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ensemblestat
###

ENSEMBLE_STAT_CONFIG_FILE = {PARM_BASE}/met_config/EnsembleStatConfig_wrapped

#LOG_ENSEMBLE_STAT_VERBOSITY = 2

#ENSEMBLE_STAT_DESC =

OBS_ENSEMBLE_STAT_WINDOW_BEGIN = -5400
OBS_ENSEMBLE_STAT_WINDOW_END = 5400

ENSEMBLE_STAT_N_MEMBERS = 2

ENSEMBLE_STAT_ENS_THRESH = 1.0
ENSEMBLE_STAT_VLD_THRESH = 1.0

ENSEMBLE_STAT_REGRID_TO_GRID = NONE

ENSEMBLE_STAT_OUTPUT_PREFIX = PYTHON


ENSEMBLE_STAT_OUTPUT_FLAG_ECNT = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_RPS = NONE
ENSEMBLE_STAT_OUTPUT_FLAG_RHIST = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_PHIST = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_ORANK = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_SSVAR = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_RELP = BOTH

ENSEMBLE_STAT_NC_ORANK_FLAG_LATLON = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_MEAN = TRUE
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_NC_ORANK_FLAG_RAW = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_RANK = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_PIT = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_VLD_COUNT = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_WEIGHT = FALSE

#ENSEMBLE_STAT_MET_OBS_ERR_TABLE =

ENSEMBLE_STAT_VERIFICATION_MASK_TEMPLATE =
    MET_BASE/poly/HMT_masks/huc4_1605_poly.nc,
    MET_BASE/poly/HMT_masks/huc4_1803_poly.nc,
    MET_BASE/poly/HMT_masks/huc4_1804_poly.nc,
    MET_BASE/poly/HMT_masks/huc4_1805_poly.nc,
    MET_BASE/poly/HMT_masks/huc4_1806_poly.nc
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *EnsembleStat MET Configuration* (page 105) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Ensemble-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
${METPLUS_MODEL}

//
// Output description to be written
```

(continues on next page)

(continued from previous page)

```
// May be set separately in each "obs.field" entry
//
${METPLUS_DESC}

//
// Output observation type to be written
//
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
${METPLUS_CENSOR_THRESH}
${METPLUS_CENSOR_VAL}
cat_thresh    = [];
nc_var_str    = "";

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}


////////////////////////////////////////////////////////////////////////////////

//prob_cat_thresh =
${METPLUS_PROB_CAT_THRESH}

//prob_pct_thresh =
${METPLUS_PROB_PCT_THRESH}

//eclv_points =
${METPLUS_ECLV_POINTS}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Forecast and observation fields to be verified
//

fcst = {

   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_ENS_THRESH}
   ${METPLUS_VLD_THRESH}
   ${METPLUS_FCST_FIELD}
}

obs = {

   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//

${METPLUS_MESSAGE_TYPE}
sid_exc        = [];
//obs_thresh     = [ NA ];
${METPLUS_OBS_THRESH}

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

${METPLUS_DUPLICATE_FLAG}
obs_summary    = NONE;
obs_perc_value = 50;
${METPLUS_SKIP_CONST}

//
```

(continues on next page)

(continued from previous page)

```
// Observation error options
// Set dist_type to NONE to use the observation error table instead
// May be set separately in each "obs.field" entry
//
obs_error = {
   ${METPLUS_OBS_ERROR_FLAG}
   dist_type        = NONE;
   dist_parm        = [];
   inst_bias_scale  = 1.0;
   inst_bias_offset = 0.0;
   min              = NA;      // Valid range of data
   max              = NA;
}

//
// Mapping of message type group name to comma-separated list of values.
//
message_type_group_map = [
   { key = "SURFACE"; val = "ADPSFC,SFCSHP,MSONET";               },
   { key = "ANYAIR";  val = "AIRCAR,AIRCFT";                      },
   { key = "ANYSFC";  val = "ADPSFC,SFCSHP,ADPUPA,PROFLR,MSONET"; },
   { key = "ONLYSF";  val = "ADPSFC,SFCSHP";                      }
];

//
// Ensemble bin sizes
// May be set separately in each "obs.field" entry
//
${METPLUS_ENS_SSVAR_BIN_SIZE}
${METPLUS_ENS_PHIST_BIN_SIZE}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}
```

(continues on next page)

(continued from previous page)

```
//
// May be set separately in each "obs.field" entry
//
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
mask = {
   ${METPLUS_MASK_GRID}
   ${METPLUS_MASK_POLY}
   sid    = [];
   llpnt = [];
}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
${METPLUS_CI_ALPHA}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
${METPLUS_OUTPUT_FLAG_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Gridded verification output types
// May be set separately in each "obs.field" entry
//
${METPLUS_NC_ORANK_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Random number generator
//
rng = {
   type = "mt19937";
   seed = "1";
}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

${METPLUS_OUTPUT_PREFIX}
//version          = "V9.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case calls a Python script to read the input data. The Python script is stored in the MET repository: /path/to/MET/installation/share/met/python/read_ascii_numpy.py

read_ascii_numpy.py

Running METplus

It is recommended to run this use case by:

Passing in EnsembleStat_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/EnsembleStat/EnsembleStat_
↪python_embedding.conf -c /path/to/user_system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/EnsembleStat/ens_python_embedding (relative to **OUTPUT_BASE**) and will contain the following files:

- ensemble_stat_PYTHON_20050807_120000V_ecnt.txt
- ensemble_stat_PYTHON_20050807_120000V_ens.nc
- ensemble_stat_PYTHON_20050807_120000V_orank.nc
- ensemble_stat_PYTHON_20050807_120000V_phist.txt
- ensemble_stat_PYTHON_20050807_120000V_relp.txt
- ensemble_stat_PYTHON_20050807_120000V_rhist.txt
- ensemble_stat_PYTHON_20050807_120000V_ssvar.txt
- ensemble_stat_PYTHON_20050807_120000V.stat

Keywords

Note:

- EnsembleStatToolUseCase
- PythonEmbeddingFileUseCase
- EnsembleAppUseCase
- ProbabilityGenerationAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-EnsembleStat.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.4 Example

7.1.32.4.1 Example: Introductory Use Case

met_tool_wrapper/Example/Example.conf

Scientific Objective

None.

Datasets

None.

METplus Components

This use case utilizes the METplus Example wrapper to demonstrate the effect of time looping and filename template METplus configuration variables.

METplus Workflow

Example is the only tool called in this example. This configuration loops by valid time every 6 hours from 2017-02-01 at 0Z until 2017-02-02 at 0Z. For each valid time, the 3, 6, 9, and 12 hour forecast leads are processed. It processes the following run times:

Valid: 2017-02-01 0Z
Forecast lead: 3 hour

Valid: 2017-02-01 0Z
Forecast lead: 6 hour

Valid: 2017-02-01 0Z
Forecast lead: 9 hour

Valid: 2017-02-01 0Z
Forecast lead: 12 hour

Valid: 2017-02-01 6Z
Forecast lead: 3 hour

Valid: 2017-02-01 6Z
Forecast lead: 6 hour

Valid: 2017-02-01 6Z
Forecast lead: 9 hour

Valid: 2017-02-01 6Z
Forecast lead: 12 hour

Valid: 2017-02-01 12Z
Forecast lead: 3 hour

Valid: 2017-02-01 12Z
Forecast lead: 6 hour

Valid: 2017-02-01 12Z
Forecast lead: 9 hour

Valid: 2017-02-01 12Z
Forecast lead: 12 hour

Valid: 2017-02-01 18Z
Forecast lead: 3 hour

Valid: 2017-02-01 18Z
Forecast lead: 6 hour

Valid: 2017-02-01 18Z
Forecast lead: 9 hour

Valid: 2017-02-01 18Z
Forecast lead: 12 hour

Valid: 2017-02-02 0Z
Forecast lead: 3 hour

Valid: 2017-02-02 0Z
Forecast lead: 6 hour

Valid: 2017-02-02 0Z
Forecast lead: 9 hour

Valid: 2017-02-02 0Z
Forecast lead: 12 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/Example/Example.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/Example/Example.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = Example


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2017020100
VALID_END = 2017020200
VALID_INCREMENT = 6H

LEAD_SEQ = 3H, 6H, 9H, 12H

EXAMPLE_CUSTOM_LOOP_LIST = ext, nc


###
# File I/O
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

EXAMPLE_INPUT_DIR = /dir/containing/example/data
EXAMPLE_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/file_{init?fmt=%Y%m%d}_{init?fmt=%H}_F{lead?fmt=
↪%3H}.{custom?fmt=%s}
```

The following configuration variables tell METplus to loop by valid time starting at 2017-02-01 0Z, ending on 2017-02-02 0Z, incrementing 6 hours each iteration:

```
LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2017020100
VALID_END = 2017020200
VALID_INCREMENT = 6H
```

The following configuration variable tells METplus to process the 3 hour, 6 hour, 9 hour, and 12 hour forecast leads for EACH valid time:

```
LEAD_SEQ = 3H, 6H, 9H, 12H
```

The following configuration variable tells METplus to look in /dir/containing/example/data to find data to process:

```
[dir]
EXAMPLE_INPUT_DIR = /dir/containing/example/data
```

Note that this variable must be found following the [dir] section header

The following configuration variable tells METplus to look for files in the input directory matching the format specified:

```
[filename_templates]
EXAMPLE_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/file_{init?fmt=%Y%m%d}_{init?fmt=%2H}_F{lead?fmt=
↪%3H}.ext
```

For example, valid time 2017-02-01 18Z and forecast lead 3 hours, the desired file is /dir/containing/example/data/20170201/file_20170201_15_F03.ext

Note that the initialization time used is 2017-02-01 15Z, which is calculated by subtracting the forecast lead from the valid time.

MET Configuration

None.

Running METplus

This use case can be run two ways:

1) Passing in Example.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/Example/Example.conf␣
↪-c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in Example.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/Example/Example.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

You should also see a series of log output listing init/valid times, forecast lead times, and filenames derived from the filename templates. Here is an excerpt:

```
12/30 19:44:02.901 metplus INFO: ****************************************
12/30 19:44:02.901 metplus INFO: * Running METplus
12/30 19:44:02.902 metplus INFO: *  at valid time: 201702010000
12/30 19:44:02.902 metplus INFO: ****************************************
12/30 19:44:02.902 metplus INFO: Running ExampleWrapper at valid time 20170201000000
12/30 19:44:02.902 metplus INFO: Input directory is /dir/containing/example/data
```

(continues on next page)

(continued from previous page)

```
12/30 19:44:02.902 metplus INFO: Input template is {init?fmt=%Y%m%d}/file_{init?fmt=%Y%m%d}_
↪{init?fmt=%2H}_F{lead?fmt=%3H}.ext
12/30 19:44:02.902 metplus INFO: Processing forecast lead 3 hours initialized at 2017-01-31␣
↪21Z and valid at 2017-02-01 00Z
12/30 19:44:02.903 metplus INFO: Looking in input directory for file: 20170131/file_20170131_
↪21_F003.ext
12/30 19:44:02.903 metplus INFO: Processing forecast lead 6 hours initialized at 2017-01-31␣
↪18Z and valid at 2017-02-01 00Z
12/30 19:44:02.903 metplus INFO: Looking in input directory for file: 20170131/file_20170131_
↪18_F006.ext
12/30 19:44:02.904 metplus INFO: Processing forecast lead 9 hours initialized at 2017-01-31␣
↪15Z and valid at 2017-02-01 00Z
12/30 19:44:02.904 metplus INFO: Looking in input directory for file: 20170131/file_20170131_
↪15_F009.ext
12/30 19:44:02.904 metplus INFO: Processing forecast lead 12 hours initialized at 2017-01-31␣
↪12Z and valid at 2017-02-01 00Z
12/30 19:44:02.904 metplus INFO: Looking in input directory for file: 20170131/file_20170131_
↪12_F012.ext
12/30 19:44:02.904 metplus INFO: ****************************************
12/30 19:44:02.904 metplus INFO: * Running METplus
12/30 19:44:02.905 metplus INFO: *  at valid time: 201702010600
12/30 19:44:02.905 metplus INFO: ****************************************
12/30 19:44:02.905 metplus INFO: Running ExampleWrapper at valid time 20170201060000
12/30 19:44:02.905 metplus INFO: Input directory is /dir/containing/example/data
12/30 19:44:02.905 metplus INFO: Input template is {init?fmt=%Y%m%d}/file_{init?fmt=%Y%m%d}_
↪{init?fmt=%2H}_F{lead?fmt=%3H}.ext
12/30 19:44:02.905 metplus INFO: Processing forecast lead 3 hours initialized at 2017-02-01␣
↪03Z and valid at 2017-02-01 06Z
12/30 19:44:02.906 metplus INFO: Looking in input directory for file: 20170201/file_20170201_
↪03_F003.ext
```

Keywords

Note:

- ExampleToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.5 ExtractTiles

7.1.32.5.1 ExtractTiles: Basic Use Case

met_tool_wrapper/ExtractTiles/ExtractTiles.conf

Scientific Objective

Read a storm stat file generated by TC-Stat and for each point on the track create an cutout of forecast and observation data valid at the track time

Datasets

Track Data: Output from TC-Stat generated from ADeck and Bdeck modified-ATCF tropical cyclone data
Forecast: GFS
Observation: GFS Analysis

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 359) section for more information.

METplus Components

This use case utilizes the METplus ExtractTiles wrapper to search for files that are valid at a given run time and generate a command to run the MET tool regrid_data_plane if all required files are found.

METplus Workflow

ExtractTiles is the only tool called in this example. It processes the following run time:

Init: 2014-12-14 0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/ExtractTiles/ExtractTiles.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/ExtractTiles/
↪ExtractTiles.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = ExtractTiles


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###


LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20141214
INIT_END = 20141214
INIT_INCREMENT = 6H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
```

(continues on next page)

(continued from previous page)

```
###

EXTRACT_TILES_TC_STAT_INPUT_DIR = {INPUT_BASE}/met_test/extract_tiles
EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/filter_{init?fmt=%Y%m%d_%H}.tcst

FCST_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/met_test/new/reduced_model_data
FCST_EXTRACT_TILES_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}
↪00_{lead?fmt=%HHH}.grb2

OBS_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/met_test/new/reduced_model_data
OBS_EXTRACT_TILES_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}
↪00_000.grb2

EXTRACT_TILES_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/ExtractTiles
FCST_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/{storm_id}/FCST_TILE_F{lead?fmt=
↪%3H}_gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%HHH}.nc
OBS_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/{storm_id}/OBS_TILE_F{lead?fmt=%3H}_
↪gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}00_000.nc

EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS = yes


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = Z2

OBS_VAR1_NAME = TMP
OBS_VAR1_LEVELS = Z2


###
# ExtractTiles Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#extracttiles
###

EXTRACT_TILES_NLAT = 60
EXTRACT_TILES_NLON = 60

EXTRACT_TILES_DLAT = 0.5
EXTRACT_TILES_DLON = 0.5
```

(continues on next page)

(continued from previous page)

```
EXTRACT_TILES_LON_ADJ = 15
EXTRACT_TILES_LAT_ADJ = 15
```

MET Configuration

None. RegridDataPlane does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in ExtractTiles.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/ExtractTiles/
↪ExtractTiles.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in ExtractTiles.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/ExtractTiles/
↪ExtractTiles.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/ExtractTiles/20141214_00 (relative to **OUTPUT_BASE**) and will contain the following files:

- ML1200942014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc
- ML1200942014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1200942014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1200942014/ANLY_TILE_F036_gfs_4_20141215_1200_000.nc
- ML1200942014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1200942014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1200942014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc
- ML1200942014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1200942014/FCST_TILE_F030_gfs_4_20141214_0000_030.nc
- ML1200942014/FCST_TILE_F036_gfs_4_20141214_0000_036.nc
- ML1200942014/FCST_TILE_F042_gfs_4_20141214_0000_042.nc
- ML1200942014/FCST_TILE_F048_gfs_4_20141214_0000_048.nc
- ML1200972014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc
- ML1200972014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1200972014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1200972014/ANLY_TILE_F036_gfs_4_20141215_1200_000.nc
- ML1200972014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1200972014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1200972014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc
- ML1200972014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1200972014/FCST_TILE_F030_gfs_4_20141214_0000_030.nc
- ML1200972014/FCST_TILE_F036_gfs_4_20141214_0000_036.nc
- ML1200972014/FCST_TILE_F042_gfs_4_20141214_0000_042.nc
- ML1200972014/FCST_TILE_F048_gfs_4_20141214_0000_048.nc
- ML1200992014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc

- ML1200992014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1200992014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1200992014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1200992014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1200992014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc
- ML1200992014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1201002014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc
- ML1201002014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1201002014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1201002014/ANLY_TILE_F036_gfs_4_20141215_1200_000.nc
- ML1201002014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1201002014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1201002014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc
- ML1201002014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1201002014/FCST_TILE_F030_gfs_4_20141214_0000_030.nc
- ML1201002014/FCST_TILE_F036_gfs_4_20141214_0000_036.nc
- ML1201032014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc
- ML1201032014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1201032014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1201032014/ANLY_TILE_F036_gfs_4_20141215_1200_000.nc
- ML1201032014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1201032014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1201032014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc
- ML1201032014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1201032014/FCST_TILE_F030_gfs_4_20141214_0000_030.nc
- ML1201032014/FCST_TILE_F036_gfs_4_20141214_0000_036.nc
- ML1201032014/FCST_TILE_F042_gfs_4_20141214_0000_042.nc
- ML1201032014/FCST_TILE_F048_gfs_4_20141214_0000_048.nc
- ML1201042014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc
- ML1201042014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1201042014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1201042014/ANLY_TILE_F036_gfs_4_20141215_1200_000.nc

- ML1201042014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1201042014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1201042014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc
- ML1201042014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1201042014/FCST_TILE_F030_gfs_4_20141214_0000_030.nc
- ML1201042014/FCST_TILE_F036_gfs_4_20141214_0000_036.nc
- ML1201042014/FCST_TILE_F042_gfs_4_20141214_0000_042.nc
- ML1201052014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc
- ML1201052014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1201052014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1201052014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1201052014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1201052014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc
- ML1201052014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1201052014/FCST_TILE_F030_gfs_4_20141214_0000_030.nc
- ML1201062014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc
- ML1201062014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1201062014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1201062014/ANLY_TILE_F036_gfs_4_20141215_1200_000.nc
- ML1201062014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1201062014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1201062014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc
- ML1201062014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1201062014/FCST_TILE_F030_gfs_4_20141214_0000_030.nc
- ML1201062014/FCST_TILE_F036_gfs_4_20141214_0000_036.nc
- ML1201072014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc
- ML1201072014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1201072014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1201072014/ANLY_TILE_F036_gfs_4_20141215_1200_000.nc
- ML1201072014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1201072014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1201072014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc

- ML1201072014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1201072014/FCST_TILE_F030_gfs_4_20141214_0000_030.nc
- ML1201072014/FCST_TILE_F036_gfs_4_20141214_0000_036.nc
- ML1201072014/FCST_TILE_F042_gfs_4_20141214_0000_042.nc
- ML1201072014/FCST_TILE_F048_gfs_4_20141214_0000_048.nc
- ML1201082014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc
- ML1201082014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1201082014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1201082014/ANLY_TILE_F036_gfs_4_20141215_1200_000.nc
- ML1201082014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1201082014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1201082014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc
- ML1201082014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1201082014/FCST_TILE_F030_gfs_4_20141214_0000_030.nc
- ML1201082014/FCST_TILE_F036_gfs_4_20141214_0000_036.nc
- ML1201082014/FCST_TILE_F042_gfs_4_20141214_0000_042.nc
- ML1201082014/FCST_TILE_F048_gfs_4_20141214_0000_048.nc
- ML1201092014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc
- ML1201092014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1201092014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1201092014/ANLY_TILE_F036_gfs_4_20141215_1200_000.nc
- ML1201092014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1201092014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1201092014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc
- ML1201092014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1201092014/FCST_TILE_F030_gfs_4_20141214_0000_030.nc
- ML1201092014/FCST_TILE_F036_gfs_4_20141214_0000_036.nc
- ML1201092014/FCST_TILE_F042_gfs_4_20141214_0000_042.nc
- ML1201102014/ANLY_TILE_F006_gfs_4_20141214_0600_000.nc
- ML1201102014/ANLY_TILE_F012_gfs_4_20141214_1200_000.nc
- ML1201102014/ANLY_TILE_F018_gfs_4_20141214_1800_000.nc
- ML1201102014/ANLY_TILE_F036_gfs_4_20141215_1200_000.nc

- ML1201102014/FCST_TILE_F006_gfs_4_20141214_0000_006.nc
- ML1201102014/FCST_TILE_F012_gfs_4_20141214_0000_012.nc
- ML1201102014/FCST_TILE_F018_gfs_4_20141214_0000_018.nc
- ML1201102014/FCST_TILE_F024_gfs_4_20141214_0000_024.nc
- ML1201102014/FCST_TILE_F030_gfs_4_20141214_0000_030.nc
- ML1201102014/FCST_TILE_F036_gfs_4_20141214_0000_036.nc

Keywords

Note:

- RegridDataPlaneToolUseCase
- GRIB2FileUseCase
- FeatureRelativeUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-ExtractTiles.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.5.2 ExtractTiles: MTD Input

met_tool_wrapper/ExtractTiles/ExtractTiles_mtd.conf

Scientific Objective

Read a MODE Time Domain (MTD) output file and use the centroid latitude and longitude values of the MTD cluster object pairs to create a cutout of forecast and observation data valid at each time.

Datasets

Track Data: Output from MODE Time Domain (MTD)
Forecast: WRF
Observation: Stage 2 NetCDF 3-hour Precipitation Accumulation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 367) section for more information.

METplus Components

This use case utilizes the METplus ExtractTiles wrapper to search for files that are valid at a given run time and generate a command to run the MET tool regrid_data_plane if all required files are found.

METplus Workflow

ExtractTiles is the only tool called in this example. It processes the following run time:

Init: 2005-08-07 0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/ExtractTiles/ExtractTiles_mtd.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/ExtractTiles/
↪ExtractTiles_mtd.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = ExtractTiles


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
```

(continues on next page)

(continued from previous page)

```
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2005080700
INIT_END = 2005080700
INIT_INCREMENT = 6H

LEAD_SEQ = 6H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

EXTRACT_TILES_MTD_INPUT_DIR = {INPUT_BASE}/met_test/new/mtd
EXTRACT_TILES_MTD_INPUT_TEMPLATE = mtd_WRF_APCP_vs_MC_PCP_APCP_03_A03_{valid?fmt=%Y%m%d_%H%M
↪%S}V_2d.txt

FCST_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst

FCST_EXTRACT_TILES_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%2H}.tm00_G212

OBS_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/met_test/new
OBS_EXTRACT_TILES_INPUT_TEMPLATE = ST2ml{valid?fmt=%Y%m%d%H}_A03h.nc

EXTRACT_TILES_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/ExtractTiles
FCST_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/FCST_TILE_F{lead?fmt=%3H}_wrfprs_
↪{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%3H}.nc
OBS_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/OBS_TILE_F{lead?fmt=%3H}_wrfprs_
↪{valid?fmt=%Y%m%d}_{valid?fmt=%H}00_000.nc

EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS = no


###
```

(continues on next page)

(continued from previous page)

```
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A03

OBS_VAR1_NAME = APCP_03
OBS_VAR1_LEVELS = "(*,*)"


###
# ExtractTiles Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#extracttiles
###

EXTRACT_TILES_NLAT = 60
EXTRACT_TILES_NLON = 60

EXTRACT_TILES_DLAT = 0.5
EXTRACT_TILES_DLON = 0.5

EXTRACT_TILES_LON_ADJ = 15
EXTRACT_TILES_LAT_ADJ = 15
```

MET Configuration

None. RegridDataPlane does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in ExtractTiles_mtd.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/ExtractTiles/
↪ExtractTiles_mtd.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in ExtractTiles_mtd.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/ExtractTiles/
↪ExtractTiles_mtd.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/ExtractTiles/20050807_00 (relative to **OUTPUT_BASE**) and will contain the following files:

- FCST_TILE_F006_wrfprs_20050807_0000_006.nc
- FCST_TILE_F009_wrfprs_20050807_0000_009.nc
- FCST_TILE_F012_wrfprs_20050807_0000_012.nc
- OBS_TILE_F006_wrfprs_20050807_0600_000.nc
- OBS_TILE_F009_wrfprs_20050807_0900_000.nc
- OBS_TILE_F012_wrfprs_20050807_1200_000.nc

Keywords

Note:

- RegridDataPlaneToolUseCase
- GRIB2FileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-ExtractTiles.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.6 GFDLTracker

7.1.32.6.1 GFDLTracker: TC Genesis Use Case

met_tool_wrapper/GFDLTracker/GFDLTracker_Genesis.conf

Scientific Objective

Setup and run GFDL Tracker applications to track cyclones in TC genesis mode. See *GFDL Tracker (optional)* (page 34) for more information. A genesis vitals file is read into the tracker. This file contains information on storms that were tracked in the previous 2 runs so that additional data is attributed to the correct storm.

Datasets

Forecast: GFS

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 377) section for more information.

METplus Components

This use case utilizes the METplus GFDLTracker wrapper to generate a command to run the GFDL Tracker Fortran applications.

METplus Workflow

GFDLTracker is the only tool called in this example. It processes the following run time:

Init: 2021-07-13 00Z
Forecast lead: All available leads (0 - 198 hour)

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/GFDLTracker/GFDLTracker_Genesis.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GFDLTracker/
↪GFDLTracker_Genesis.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###
PROCESS_LIST = GFDLTracker


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT

INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2021071300
INIT_END = 2021071300
INIT_INCREMENT = 6H

LEAD_SEQ = *


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

GFDL_TRACKER_INPUT_DIR = {INPUT_BASE}/met_test/gfdl/gfs
GFDL_TRACKER_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/gfs.t{init?fmt=%H}z.pgrb2.1p00.f{lead?fmt=
↪%3H}

GFDL_TRACKER_TC_VITALS_INPUT_DIR = {GFDL_TRACKER_INPUT_DIR}
GFDL_TRACKER_TC_VITALS_INPUT_TEMPLATE = syndat_tcvitals.{init?fmt=%Y}

GFDL_TRACKER_GEN_VITALS_INPUT_DIR = {GFDL_TRACKER_INPUT_DIR}
GFDL_TRACKER_GEN_VITALS_INPUT_TEMPLATE = genesis.vitals.gfso.glbl.{init?fmt=%Y%m}

GFDL_TRACKER_OUTPUT_DIR = {OUTPUT_BASE}/gfdl_tracker/genesis
GFDL_TRACKER_OUTPUT_TEMPLATE = gfs.{init?fmt=%Y%m%d%H}.genesis.txt


###
# GFDLTracker Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gfdltracker
###

GFDL_TRACKER_GRIB_VERSION = 2

GFDL_TRACKER_NML_TEMPLATE_FILE = {PARM_BASE}/use_cases/met_tool_wrapper/GFDLTracker/template.
↪nml

GFDL_TRACKER_DATEIN_INP_MODEL = 1
GFDL_TRACKER_DATEIN_INP_MODTYP = "global"
GFDL_TRACKER_DATEIN_INP_LT_UNITS = "hours"
GFDL_TRACKER_DATEIN_INP_FILE_SEQ = "multi"
GFDL_TRACKER_DATEIN_INP_NESTTYP = "fixed"

GFDL_TRACKER_ATCFINFO_ATCFNUM = 81
GFDL_TRACKER_ATCFINFO_ATCFNAME = "GFML"
GFDL_TRACKER_ATCFINFO_ATCFFREQ = 600

GFDL_TRACKER_TRACKERINFO_TYPE = "tcgen"
GFDL_TRACKER_TRACKERINFO_MSLPTHRESH = 0.0015
GFDL_TRACKER_TRACKERINFO_USE_BACKUP_MSLP_GRAD_CHECK = True
GFDL_TRACKER_TRACKERINFO_V850THRESH = 1.5
GFDL_TRACKER_TRACKERINFO_USE_BACKUP_850_VT_CHECK = True
GFDL_TRACKER_TRACKERINFO_ENABLE_TIMING = 1
GFDL_TRACKER_TRACKERINFO_GRIDTYPE = "global"
GFDL_TRACKER_TRACKERINFO_CONTINT = 100.0
```

(continues on next page)

(continued from previous page)

```
GFDL_TRACKER_TRACKERINFO_WANT_OCI = T
GFDL_TRACKER_TRACKERINFO_OUT_VIT = True
GFDL_TRACKER_TRACKERINFO_USE_LAND_MASK = False
GFDL_TRACKER_TRACKERINFO_INP_DATA_TYPE = "grib"
GFDL_TRACKER_TRACKERINFO_GRIBVER = 2
GFDL_TRACKER_TRACKERINFO_G2_JPDTN = 0
GFDL_TRACKER_TRACKERINFO_G2_MSLP_PARM_ID = 1
GFDL_TRACKER_TRACKERINFO_G1_MSLP_PARM_ID = 2
GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_TYP = 105
GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_VAL = 10

GFDL_TRACKER_TRACKERINFO_WESTBD = 0
GFDL_TRACKER_TRACKERINFO_EASTBD = 358
GFDL_TRACKER_TRACKERINFO_SOUTHBD = -89
GFDL_TRACKER_TRACKERINFO_NORTHBD = 89

GFDL_TRACKER_PHASEINFO_PHASEFLAG = True
GFDL_TRACKER_PHASEINFO_PHASESCHEME = "both"
GFDL_TRACKER_PHASEINFO_WCORE_DEPTH = 1.0

GFDL_TRACKER_STRUCTINFO_STRUCTFLAG = False
GFDL_TRACKER_STRUCTINFO_IKEFLAG = False

GFDL_TRACKER_FNAMEINFO_GMODNAME = "gfs"
GFDL_TRACKER_FNAMEINFO_RUNDESCR = "t{init?fmt=%H}z.pgrb2"
GFDL_TRACKER_FNAMEINFO_ATCFDESCR = "1p00"

GFDL_TRACKER_WAITINFO_USE_WAITFOR = True
GFDL_TRACKER_WAITINFO_WAIT_MIN_AGE = 10
GFDL_TRACKER_WAITINFO_WAIT_MIN_SIZE = 100
GFDL_TRACKER_WAITINFO_WAIT_MAX_WAIT = 3600
GFDL_TRACKER_WAITINFO_WAIT_SLEEPTIME = 5
GFDL_TRACKER_WAITINFO_USE_PER_FCST_COMMAND = True
GFDL_TRACKER_WAITINFO_PER_FCST_COMMAND = "./deliver %[FHOUR] %[FMIN]"

GFDL_TRACKER_NETCDFINFO_LAT_NAME = ""
GFDL_TRACKER_NETCDFINFO_LMASKNAME = ""
GFDL_TRACKER_NETCDFINFO_LON_NAME = ""
GFDL_TRACKER_NETCDFINFO_MSLPNAME = ""
GFDL_TRACKER_NETCDFINFO_NETCDF_FILENAME = ""
GFDL_TRACKER_NETCDFINFO_NUM_NETCDF_VARS = 0
GFDL_TRACKER_NETCDFINFO_RV700NAME = ""
GFDL_TRACKER_NETCDFINFO_RV850NAME = ""
GFDL_TRACKER_NETCDFINFO_TIME_NAME = ""
GFDL_TRACKER_NETCDFINFO_TIME_UNITS = ""
```

(continues on next page)

(continued from previous page)

```
GFDL_TRACKER_NETCDFINFO_TMEAN_300_500_NAME = ""
GFDL_TRACKER_NETCDFINFO_U500NAME = ""
GFDL_TRACKER_NETCDFINFO_U700NAME = ""
GFDL_TRACKER_NETCDFINFO_U850NAME = ""
GFDL_TRACKER_NETCDFINFO_USFCNAME = ""
GFDL_TRACKER_NETCDFINFO_V500NAME = ""
GFDL_TRACKER_NETCDFINFO_V700NAME = ""
GFDL_TRACKER_NETCDFINFO_V850NAME = ""
GFDL_TRACKER_NETCDFINFO_VSFCNAME = ""
GFDL_TRACKER_NETCDFINFO_Z200NAME = ""
GFDL_TRACKER_NETCDFINFO_Z300NAME = ""
GFDL_TRACKER_NETCDFINFO_Z350NAME = ""
GFDL_TRACKER_NETCDFINFO_Z400NAME = ""
GFDL_TRACKER_NETCDFINFO_Z450NAME = ""
GFDL_TRACKER_NETCDFINFO_Z500NAME = ""
GFDL_TRACKER_NETCDFINFO_Z550NAME = ""
GFDL_TRACKER_NETCDFINFO_Z600NAME = ""
GFDL_TRACKER_NETCDFINFO_Z650NAME = ""
GFDL_TRACKER_NETCDFINFO_Z700NAME = ""
GFDL_TRACKER_NETCDFINFO_Z750NAME = ""
GFDL_TRACKER_NETCDFINFO_Z800NAME = ""
GFDL_TRACKER_NETCDFINFO_Z850NAME = ""
GFDL_TRACKER_NETCDFINFO_Z900NAME = ""

GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA850 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA700 = False
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC850 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC700 = False
GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH850 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH700 = False
GFDL_TRACKER_USER_WANTS_TO_TRACK_MSLP = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRCSFC = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETASFC = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK500850 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200500 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200850 = True

GFDL_TRACKER_VERBOSE_VERB = 3
GFDL_TRACKER_VERBOSE_VERB_G2 = 0
```

GFDL Tracker Configuration

METplus replaces values in the template configuration files read by the tracker based on user settings in the METplus configuration file.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

```
&datein
  inp%bcc = ${METPLUS_DATEIN_INP_BCC},
  inp%byy = ${METPLUS_DATEIN_INP_BYY},
  inp%bmm = ${METPLUS_DATEIN_INP_BMM},
  inp%bdd = ${METPLUS_DATEIN_INP_BDD},
  inp%bhh = ${METPLUS_DATEIN_INP_BHH},
  inp%model = ${METPLUS_DATEIN_INP_MODEL},
  inp%modtyp = ${METPLUS_DATEIN_INP_MODTYP},
  inp%lt_units = ${METPLUS_DATEIN_INP_LT_UNITS},
  inp%file_seq = ${METPLUS_DATEIN_INP_FILE_SEQ},
  inp%nesttyp = ${METPLUS_DATEIN_INP_NESTTYP},
/

&atcfinfo
  atcfnum = ${METPLUS_ATCFINFO_ATCFNUM},
  atcfname = ${METPLUS_ATCFINFO_ATCFNAME},
  atcfymdh = ${METPLUS_ATCFINFO_ATCFYMDH},
  atcffreq = ${METPLUS_ATCFINFO_ATCFFREQ},
/

&trackerinfo
  trkrinfo%type = ${METPLUS_TRACKERINFO_TYPE},
  trkrinfo%mslpthresh = ${METPLUS_TRACKERINFO_MSLPTHRESH},
  trkrinfo%use_backup_mslp_grad_check = ${METPLUS_TRACKERINFO_USE_BACKUP_MSLP_GRAD_CHECK},
  trkrinfo%v850thresh = ${METPLUS_TRACKERINFO_V850THRESH},
  trkrinfo%use_backup_850_vt_check = ${METPLUS_TRACKERINFO_USE_BACKUP_850_VT_CHECK},
  trkrinfo%enable_timing = ${METPLUS_TRACKERINFO_ENABLE_TIMING},
  trkrinfo%gridtype = ${METPLUS_TRACKERINFO_GRIDTYPE},
  trkrinfo%contint = ${METPLUS_TRACKERINFO_CONTINT},
  trkrinfo%want_oci = ${METPLUS_TRACKERINFO_WANT_OCI},
  trkrinfo%out_vit = ${METPLUS_TRACKERINFO_OUT_VIT},
  trkrinfo%use_land_mask = ${METPLUS_TRACKERINFO_USE_LAND_MASK},
  trkrinfo%inp_data_type = ${METPLUS_TRACKERINFO_INP_DATA_TYPE},
  trkrinfo%gribver = ${METPLUS_TRACKERINFO_GRIBVER},
  trkrinfo%g2_jpdtn = ${METPLUS_TRACKERINFO_G2_JPDTN},
  trkrinfo%g2_mslp_parm_id = ${METPLUS_TRACKERINFO_G2_MSLP_PARM_ID},
  trkrinfo%g1_mslp_parm_id = ${METPLUS_TRACKERINFO_G1_MSLP_PARM_ID},
  trkrinfo%g1_sfcwind_lev_typ = ${METPLUS_TRACKERINFO_G1_SFCWIND_LEV_TYP},
  trkrinfo%g1_sfcwind_lev_val = ${METPLUS_TRACKERINFO_G1_SFCWIND_LEV_VAL},
```

(continues on next page)

(continued from previous page)

```
  trkrinfo%westbd = ${METPLUS_TRACKERINFO_WESTBD},
  trkrinfo%eastbd = ${METPLUS_TRACKERINFO_EASTBD},
  trkrinfo%southbd = ${METPLUS_TRACKERINFO_SOUTHBD},
  trkrinfo%northbd = ${METPLUS_TRACKERINFO_NORTHBD},
/

&phaseinfo
  phaseflag = ${METPLUS_PHASEINFO_PHASEFLAG},
  phasescheme = ${METPLUS_PHASEINFO_PHASESCHEME},
  wcore_depth = ${METPLUS_PHASEINFO_WCORE_DEPTH},
/

&structinfo
  structflag = ${METPLUS_STRUCTINFO_STRUCTFLAG},
  ikeflag = ${METPLUS_STRUCTINFO_IKEFLAG},
/

&fnameinfo
  gmodname = ${METPLUS_FNAMEINFO_GMODNAME},
  rundescr = ${METPLUS_FNAMEINFO_RUNDESCR},
  atcfdescr = ${METPLUS_FNAMEINFO_ATCFDESCR},
/

&waitinfo
  use_waitfor = ${METPLUS_WAITINFO_USE_WAITFOR},
  wait_min_age = ${METPLUS_WAITINFO_WAIT_MIN_AGE},
  wait_min_size = ${METPLUS_WAITINFO_WAIT_MIN_SIZE},
  wait_max_wait = ${METPLUS_WAITINFO_WAIT_MAX_WAIT},
  wait_sleeptime = ${METPLUS_WAITINFO_WAIT_SLEEPTIME},
  use_per_fcst_command = ${METPLUS_WAITINFO_USE_PER_FCST_COMMAND},
  per_fcst_command = ${METPLUS_WAITINFO_PER_FCST_COMMAND},
/

&netcdflist
  netcdfinfo%lat_name = ${METPLUS_NETCDFINFO_LAT_NAME},
  netcdfinfo%lmaskname = ${METPLUS_NETCDFINFO_LMASKNAME},
  netcdfinfo%lon_name = ${METPLUS_NETCDFINFO_LON_NAME},
  netcdfinfo%mslpname = ${METPLUS_NETCDFINFO_MSLPNAME},
  netcdfinfo%netcdf_filename = ${METPLUS_NETCDFINFO_NETCDF_FILENAME},
  netcdfinfo%num_netcdf_vars = ${METPLUS_NETCDFINFO_NUM_NETCDF_VARS},
  netcdfinfo%rv700name = ${METPLUS_NETCDFINFO_RV700NAME},
  netcdfinfo%rv850name = ${METPLUS_NETCDFINFO_RV850NAME},
  netcdfinfo%time_name = ${METPLUS_NETCDFINFO_TIME_NAME},
  netcdfinfo%time_units = ${METPLUS_NETCDFINFO_TIME_UNITS},
  netcdfinfo%tmean_300_500_name = ${METPLUS_NETCDFINFO_TMEAN_300_500_NAME},
```

(continues on next page)

(continued from previous page)

```
  netcdfinfo%u500name = ${METPLUS_NETCDFINFO_U500NAME},
  netcdfinfo%u700name = ${METPLUS_NETCDFINFO_U700NAME},
  netcdfinfo%u850name = ${METPLUS_NETCDFINFO_U850NAME},
  netcdfinfo%usfcname = ${METPLUS_NETCDFINFO_USFCNAME},
  netcdfinfo%v500name = ${METPLUS_NETCDFINFO_V500NAME},
  netcdfinfo%v700name = ${METPLUS_NETCDFINFO_V700NAME},
  netcdfinfo%v850name = ${METPLUS_NETCDFINFO_V850NAME},
  netcdfinfo%vsfcname = ${METPLUS_NETCDFINFO_VSFCNAME},
  netcdfinfo%z200name = ${METPLUS_NETCDFINFO_Z200NAME},
  netcdfinfo%z300name = ${METPLUS_NETCDFINFO_Z300NAME},
  netcdfinfo%z350name = ${METPLUS_NETCDFINFO_Z350NAME},
  netcdfinfo%z400name = ${METPLUS_NETCDFINFO_Z400NAME},
  netcdfinfo%z450name = ${METPLUS_NETCDFINFO_Z450NAME},
  netcdfinfo%z500name = ${METPLUS_NETCDFINFO_Z500NAME},
  netcdfinfo%z550name = ${METPLUS_NETCDFINFO_Z550NAME},
  netcdfinfo%z600name = ${METPLUS_NETCDFINFO_Z600NAME},
  netcdfinfo%z650name = ${METPLUS_NETCDFINFO_Z650NAME},
  netcdfinfo%z700name = ${METPLUS_NETCDFINFO_Z700NAME},
  netcdfinfo%z750name = ${METPLUS_NETCDFINFO_Z750NAME},
  netcdfinfo%z800name = ${METPLUS_NETCDFINFO_Z800NAME},
  netcdfinfo%z850name = ${METPLUS_NETCDFINFO_Z850NAME},
  netcdfinfo%z900name = ${METPLUS_NETCDFINFO_Z900NAME},
/

&parmpreflist
  user_wants_to_track_zeta700 = ${METPLUS_USER_WANTS_TO_TRACK_ZETA700},
  user_wants_to_track_wcirc850 = ${METPLUS_USER_WANTS_TO_TRACK_WCIRC850},
  user_wants_to_track_wcirc700 = ${METPLUS_USER_WANTS_TO_TRACK_WCIRC700},
  user_wants_to_track_gph850 = ${METPLUS_USER_WANTS_TO_TRACK_GPH850},
  user_wants_to_track_gph700 = ${METPLUS_USER_WANTS_TO_TRACK_GPH700},
  user_wants_to_track_mslp = ${METPLUS_USER_WANTS_TO_TRACK_MSLP},
  user_wants_to_track_wcircsfc = ${METPLUS_USER_WANTS_TO_TRACK_WCIRCSFC},
  user_wants_to_track_zetasfc = ${METPLUS_USER_WANTS_TO_TRACK_ZETASFC},
  user_wants_to_track_thick500850 = ${METPLUS_USER_WANTS_TO_TRACK_THICK500850},
  user_wants_to_track_thick200500 = ${METPLUS_USER_WANTS_TO_TRACK_THICK200500},
  user_wants_to_track_thick200850 = ${METPLUS_USER_WANTS_TO_TRACK_THICK200850},
  user_wants_to_track_zeta850 = ${METPLUS_USER_WANTS_TO_TRACK_ZETA850},
/

&verbose
  verb = ${METPLUS_VERBOSE_VERB},
  verb_g2 = ${METPLUS_VERBOSE_VERB_G2},
/
```

Running METplus

This use case can be run by passing in the conf file to the run script:

```
run_metplus.py /path/to/METplus/parm/use_cases/met_tool_wrapper/GFDLTracker/GFDLTracker_
↪Genesis.conf
```

See the *Running METplus* (page 26) section of the User's Guide for more information on how to run use cases.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in gfdl_tracker/genesis (relative to **OUTPUT_BASE**) and will contain the following file:

- gfs.2021071300.genesis.txt
- input.202107130000.nml

Keywords

Note:

- GFDLTrackerToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.6.2 GFDLTracker: Tropical Cyclone Use Case

met_tool_wrapper/GFDLTracker/GFDLTracker_TC.conf

Scientific Objective

Setup and run GFDL Tracker applications to track tropical cyclones. See *GFDL Tracker (optional)* (page 34) for more information.

Datasets

Forecast: HWRF

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 385) section for more information.

METplus Components

This use case utilizes the METplus GFDLTracker wrapper to generate a command to run the GFDL Tracker Fortran applications.

METplus Workflow

GFDLTracker is the only tool called in this example. It processes the following run time:

Init: 2016-09-06 00Z
Forecast lead: All available leads (0 - 126 hour)

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/GFDLTracker/GFDLTracker_TC.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GFDLTracker/
↪GFDLTracker_TC.html

```

(continues on next page)

(continued from previous page)

```
# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GFDLTracker


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT

INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2016090600
INIT_END = 2016090600
LEAD_SEQ = *
#LEAD_SEQ = begin_end_incr(0, 18, 6)H
#LEAD_SEQ = begin_end_incr(0, 9, 1)H, begin_end_incr(12,126,3)H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

GFDL_TRACKER_INPUT_DIR = {INPUT_BASE}/met_test/gfdl/hwrf
GFDL_TRACKER_INPUT_TEMPLATE = hwrf.25x25.EP152016.{init?fmt=%Y%m%d%H}.f{lead?fmt=%5M}

GFDL_TRACKER_TC_VITALS_INPUT_DIR = {GFDL_TRACKER_INPUT_DIR}
GFDL_TRACKER_TC_VITALS_INPUT_TEMPLATE = tcvit_rsmc_storms.txt
```

(continues on next page)

(continued from previous page)

```
GFDL_TRACKER_OUTPUT_DIR = {OUTPUT_BASE}/gfdl_tracker/tc
GFDL_TRACKER_OUTPUT_TEMPLATE = hwrf.{init?fmt=%Y%m%d%H}.track.txt


###
# GFDLTracker Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gfdltracker
###

GFDL_TRACKER_GRIB_VERSION = 1

GFDL_TRACKER_NML_TEMPLATE_FILE = {PARM_BASE}/use_cases/met_tool_wrapper/GFDLTracker/template.
↪nml

GFDL_TRACKER_DATEIN_INP_MODEL = 17
GFDL_TRACKER_DATEIN_INP_MODTYP = "regional"
GFDL_TRACKER_DATEIN_INP_LT_UNITS = "hours"
GFDL_TRACKER_DATEIN_INP_FILE_SEQ = "multi"
GFDL_TRACKER_DATEIN_INP_NESTTYP = "moveable"

GFDL_TRACKER_ATCFINFO_ATCFNUM = 81
GFDL_TRACKER_ATCFINFO_ATCFNAME = "HWRF"
GFDL_TRACKER_ATCFINFO_ATCFFREQ = 100

GFDL_TRACKER_TRACKERINFO_TYPE = "tracker"
GFDL_TRACKER_TRACKERINFO_MSLPTHRESH = 0.0015
GFDL_TRACKER_TRACKERINFO_USE_BACKUP_MSLP_GRAD_CHECK = True
GFDL_TRACKER_TRACKERINFO_V850THRESH = 1.5
GFDL_TRACKER_TRACKERINFO_USE_BACKUP_850_VT_CHECK = True
GFDL_TRACKER_TRACKERINFO_ENABLE_TIMING = 1
GFDL_TRACKER_TRACKERINFO_GRIDTYPE = "regional"
GFDL_TRACKER_TRACKERINFO_CONTINT = 100.0
GFDL_TRACKER_TRACKERINFO_WANT_OCI = T
GFDL_TRACKER_TRACKERINFO_OUT_VIT = True
GFDL_TRACKER_TRACKERINFO_USE_LAND_MASK = True
GFDL_TRACKER_TRACKERINFO_INP_DATA_TYPE = "grib"
GFDL_TRACKER_TRACKERINFO_GRIBVER = 1
GFDL_TRACKER_TRACKERINFO_G2_JPDTN = 0
GFDL_TRACKER_TRACKERINFO_G2_MSLP_PARM_ID = 192
GFDL_TRACKER_TRACKERINFO_G1_MSLP_PARM_ID = 2
GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_TYP = 105
GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_VAL = 10

GFDL_TRACKER_PHASEINFO_PHASEFLAG = True
GFDL_TRACKER_PHASEINFO_PHASESCHEME = "both"
```

(continues on next page)

(continued from previous page)

```
GFDL_TRACKER_PHASEINFO_WCORE_DEPTH = 1.0

GFDL_TRACKER_STRUCTINFO_STRUCTFLAG = False
GFDL_TRACKER_STRUCTINFO_IKEFLAG = False

GFDL_TRACKER_FNAMEINFO_GMODNAME = "hwrf"
GFDL_TRACKER_FNAMEINFO_RUNDESCR = "25x25"
GFDL_TRACKER_FNAMEINFO_ATCFDESCR = "EP152016"

GFDL_TRACKER_WAITINFO_USE_WAITFOR = True
GFDL_TRACKER_WAITINFO_WAIT_MIN_AGE = 10
GFDL_TRACKER_WAITINFO_WAIT_MIN_SIZE = 100
GFDL_TRACKER_WAITINFO_WAIT_MAX_WAIT = 3600
GFDL_TRACKER_WAITINFO_WAIT_SLEEPTIME = 5
GFDL_TRACKER_WAITINFO_USE_PER_FCST_COMMAND = True
GFDL_TRACKER_WAITINFO_PER_FCST_COMMAND = "./deliver %[FHOUR] %[FMIN]"

GFDL_TRACKER_NETCDFINFO_LAT_NAME = ""
GFDL_TRACKER_NETCDFINFO_LMASKNAME = ""
GFDL_TRACKER_NETCDFINFO_LON_NAME = ""
GFDL_TRACKER_NETCDFINFO_MSLPNAME = ""
GFDL_TRACKER_NETCDFINFO_NETCDF_FILENAME = ""
GFDL_TRACKER_NETCDFINFO_NUM_NETCDF_VARS = 0
GFDL_TRACKER_NETCDFINFO_RV700NAME = ""
GFDL_TRACKER_NETCDFINFO_RV850NAME = ""
GFDL_TRACKER_NETCDFINFO_TIME_NAME = ""
GFDL_TRACKER_NETCDFINFO_TIME_UNITS = ""
GFDL_TRACKER_NETCDFINFO_TMEAN_300_500_NAME = ""
GFDL_TRACKER_NETCDFINFO_U500NAME = ""
GFDL_TRACKER_NETCDFINFO_U700NAME = ""
GFDL_TRACKER_NETCDFINFO_U850NAME = ""
GFDL_TRACKER_NETCDFINFO_USFCNAME = ""
GFDL_TRACKER_NETCDFINFO_V500NAME = ""
GFDL_TRACKER_NETCDFINFO_V700NAME = ""
GFDL_TRACKER_NETCDFINFO_V850NAME = ""
GFDL_TRACKER_NETCDFINFO_VSFCNAME = ""
GFDL_TRACKER_NETCDFINFO_Z200NAME = ""
GFDL_TRACKER_NETCDFINFO_Z300NAME = ""
GFDL_TRACKER_NETCDFINFO_Z350NAME = ""
GFDL_TRACKER_NETCDFINFO_Z400NAME = ""
GFDL_TRACKER_NETCDFINFO_Z450NAME = ""
GFDL_TRACKER_NETCDFINFO_Z500NAME = ""
GFDL_TRACKER_NETCDFINFO_Z550NAME = ""
GFDL_TRACKER_NETCDFINFO_Z600NAME = ""
GFDL_TRACKER_NETCDFINFO_Z650NAME = ""
```

(continues on next page)

(continued from previous page)

```
GFDL_TRACKER_NETCDFINFO_Z700NAME = ""
GFDL_TRACKER_NETCDFINFO_Z750NAME = ""
GFDL_TRACKER_NETCDFINFO_Z800NAME = ""
GFDL_TRACKER_NETCDFINFO_Z850NAME = ""
GFDL_TRACKER_NETCDFINFO_Z900NAME = ""

GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA700 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC850 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC700 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH850 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH700 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_MSLP = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRCSFC = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETASFC = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK500850 = False
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200500 = False
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200850 = False
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA850 = True

GFDL_TRACKER_VERBOSE_VERB = 3
GFDL_TRACKER_VERBOSE_VERB_G2 = 0
```

GFDL Tracker Configuration

METplus replaces values in the template configuration files read by the tracker based on user settings in the METplus configuration file.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

```
&datein
  inp%bcc = ${METPLUS_DATEIN_INP_BCC},
  inp%byy = ${METPLUS_DATEIN_INP_BYY},
  inp%bmm = ${METPLUS_DATEIN_INP_BMM},
  inp%bdd = ${METPLUS_DATEIN_INP_BDD},
  inp%bhh = ${METPLUS_DATEIN_INP_BHH},
  inp%model = ${METPLUS_DATEIN_INP_MODEL},
  inp%modtyp = ${METPLUS_DATEIN_INP_MODTYP},
  inp%lt_units = ${METPLUS_DATEIN_INP_LT_UNITS},
  inp%file_seq = ${METPLUS_DATEIN_INP_FILE_SEQ},
  inp%nesttyp = ${METPLUS_DATEIN_INP_NESTTYP},
/

&atcfinfo
  atcfnum = ${METPLUS_ATCFINFO_ATCFNUM},
```

(continues on next page)

(continued from previous page)

```
  atcfname = ${METPLUS_ATCFINFO_ATCFNAME},
  atcfymdh = ${METPLUS_ATCFINFO_ATCFYMDH},
  atcffreq = ${METPLUS_ATCFINFO_ATCFFREQ},
/

&trackerinfo
  trkrinfo%type = ${METPLUS_TRACKERINFO_TYPE},
  trkrinfo%mslpthresh = ${METPLUS_TRACKERINFO_MSLPTHRESH},
  trkrinfo%use_backup_mslp_grad_check = ${METPLUS_TRACKERINFO_USE_BACKUP_MSLP_GRAD_CHECK},
  trkrinfo%v850thresh = ${METPLUS_TRACKERINFO_V850THRESH},
  trkrinfo%use_backup_850_vt_check = ${METPLUS_TRACKERINFO_USE_BACKUP_850_VT_CHECK},
  trkrinfo%enable_timing = ${METPLUS_TRACKERINFO_ENABLE_TIMING},
  trkrinfo%gridtype = ${METPLUS_TRACKERINFO_GRIDTYPE},
  trkrinfo%contint = ${METPLUS_TRACKERINFO_CONTINT},
  trkrinfo%want_oci = ${METPLUS_TRACKERINFO_WANT_OCI},
  trkrinfo%out_vit = ${METPLUS_TRACKERINFO_OUT_VIT},
  trkrinfo%use_land_mask = ${METPLUS_TRACKERINFO_USE_LAND_MASK},
  trkrinfo%inp_data_type = ${METPLUS_TRACKERINFO_INP_DATA_TYPE},
  trkrinfo%gribver = ${METPLUS_TRACKERINFO_GRIBVER},
  trkrinfo%g2_jpdtn = ${METPLUS_TRACKERINFO_G2_JPDTN},
  trkrinfo%g2_mslp_parm_id = ${METPLUS_TRACKERINFO_G2_MSLP_PARM_ID},
  trkrinfo%g1_mslp_parm_id = ${METPLUS_TRACKERINFO_G1_MSLP_PARM_ID},
  trkrinfo%g1_sfcwind_lev_typ = ${METPLUS_TRACKERINFO_G1_SFCWIND_LEV_TYP},
  trkrinfo%g1_sfcwind_lev_val = ${METPLUS_TRACKERINFO_G1_SFCWIND_LEV_VAL},
  trkrinfo%westbd = ${METPLUS_TRACKERINFO_WESTBD},
  trkrinfo%eastbd = ${METPLUS_TRACKERINFO_EASTBD},
  trkrinfo%southbd = ${METPLUS_TRACKERINFO_SOUTHBD},
  trkrinfo%northbd = ${METPLUS_TRACKERINFO_NORTHBD},
/

&phaseinfo
  phaseflag = ${METPLUS_PHASEINFO_PHASEFLAG},
  phasescheme = ${METPLUS_PHASEINFO_PHASESCHEME},
  wcore_depth = ${METPLUS_PHASEINFO_WCORE_DEPTH},
/

&structinfo
  structflag = ${METPLUS_STRUCTINFO_STRUCTFLAG},
  ikeflag = ${METPLUS_STRUCTINFO_IKEFLAG},
/

&fnameinfo
  gmodname = ${METPLUS_FNAMEINFO_GMODNAME},
  rundescr = ${METPLUS_FNAMEINFO_RUNDESCR},
  atcfdescr = ${METPLUS_FNAMEINFO_ATCFDESCR},
```

(continues on next page)

(continued from previous page)

```
/

&waitinfo
  use_waitfor = ${METPLUS_WAITINFO_USE_WAITFOR},
  wait_min_age = ${METPLUS_WAITINFO_WAIT_MIN_AGE},
  wait_min_size = ${METPLUS_WAITINFO_WAIT_MIN_SIZE},
  wait_max_wait = ${METPLUS_WAITINFO_WAIT_MAX_WAIT},
  wait_sleeptime = ${METPLUS_WAITINFO_WAIT_SLEEPTIME},
  use_per_fcst_command = ${METPLUS_WAITINFO_USE_PER_FCST_COMMAND},
  per_fcst_command = ${METPLUS_WAITINFO_PER_FCST_COMMAND},
/

&netcdflist
  netcdfinfo%lat_name = ${METPLUS_NETCDFINFO_LAT_NAME},
  netcdfinfo%lmaskname = ${METPLUS_NETCDFINFO_LMASKNAME},
  netcdfinfo%lon_name = ${METPLUS_NETCDFINFO_LON_NAME},
  netcdfinfo%mslpname = ${METPLUS_NETCDFINFO_MSLPNAME},
  netcdfinfo%netcdf_filename = ${METPLUS_NETCDFINFO_NETCDF_FILENAME},
  netcdfinfo%num_netcdf_vars = ${METPLUS_NETCDFINFO_NUM_NETCDF_VARS},
  netcdfinfo%rv700name = ${METPLUS_NETCDFINFO_RV700NAME},
  netcdfinfo%rv850name = ${METPLUS_NETCDFINFO_RV850NAME},
  netcdfinfo%time_name = ${METPLUS_NETCDFINFO_TIME_NAME},
  netcdfinfo%time_units = ${METPLUS_NETCDFINFO_TIME_UNITS},
  netcdfinfo%tmean_300_500_name = ${METPLUS_NETCDFINFO_TMEAN_300_500_NAME},
  netcdfinfo%u500name = ${METPLUS_NETCDFINFO_U500NAME},
  netcdfinfo%u700name = ${METPLUS_NETCDFINFO_U700NAME},
  netcdfinfo%u850name = ${METPLUS_NETCDFINFO_U850NAME},
  netcdfinfo%usfcname = ${METPLUS_NETCDFINFO_USFCNAME},
  netcdfinfo%v500name = ${METPLUS_NETCDFINFO_V500NAME},
  netcdfinfo%v700name = ${METPLUS_NETCDFINFO_V700NAME},
  netcdfinfo%v850name = ${METPLUS_NETCDFINFO_V850NAME},
  netcdfinfo%vsfcname = ${METPLUS_NETCDFINFO_VSFCNAME},
  netcdfinfo%z200name = ${METPLUS_NETCDFINFO_Z200NAME},
  netcdfinfo%z300name = ${METPLUS_NETCDFINFO_Z300NAME},
  netcdfinfo%z350name = ${METPLUS_NETCDFINFO_Z350NAME},
  netcdfinfo%z400name = ${METPLUS_NETCDFINFO_Z400NAME},
  netcdfinfo%z450name = ${METPLUS_NETCDFINFO_Z450NAME},
  netcdfinfo%z500name = ${METPLUS_NETCDFINFO_Z500NAME},
  netcdfinfo%z550name = ${METPLUS_NETCDFINFO_Z550NAME},
  netcdfinfo%z600name = ${METPLUS_NETCDFINFO_Z600NAME},
  netcdfinfo%z650name = ${METPLUS_NETCDFINFO_Z650NAME},
  netcdfinfo%z700name = ${METPLUS_NETCDFINFO_Z700NAME},
  netcdfinfo%z750name = ${METPLUS_NETCDFINFO_Z750NAME},
  netcdfinfo%z800name = ${METPLUS_NETCDFINFO_Z800NAME},
  netcdfinfo%z850name = ${METPLUS_NETCDFINFO_Z850NAME},
```

(continues on next page)

(continued from previous page)

```
  netcdfinfo%z900name = ${METPLUS_NETCDFINFO_Z900NAME},
/

&parmpreflist
  user_wants_to_track_zeta700 = ${METPLUS_USER_WANTS_TO_TRACK_ZETA700},
  user_wants_to_track_wcirc850 = ${METPLUS_USER_WANTS_TO_TRACK_WCIRC850},
  user_wants_to_track_wcirc700 = ${METPLUS_USER_WANTS_TO_TRACK_WCIRC700},
  user_wants_to_track_gph850 = ${METPLUS_USER_WANTS_TO_TRACK_GPH850},
  user_wants_to_track_gph700 = ${METPLUS_USER_WANTS_TO_TRACK_GPH700},
  user_wants_to_track_mslp = ${METPLUS_USER_WANTS_TO_TRACK_MSLP},
  user_wants_to_track_wcircsfc = ${METPLUS_USER_WANTS_TO_TRACK_WCIRCSFC},
  user_wants_to_track_zetasfc = ${METPLUS_USER_WANTS_TO_TRACK_ZETASFC},
  user_wants_to_track_thick500850 = ${METPLUS_USER_WANTS_TO_TRACK_THICK500850},
  user_wants_to_track_thick200500 = ${METPLUS_USER_WANTS_TO_TRACK_THICK200500},
  user_wants_to_track_thick200850 = ${METPLUS_USER_WANTS_TO_TRACK_THICK200850},
  user_wants_to_track_zeta850 = ${METPLUS_USER_WANTS_TO_TRACK_ZETA850},
/

&verbose
  verb = ${METPLUS_VERBOSE_VERB},
  verb_g2 = ${METPLUS_VERBOSE_VERB_G2},
/
```

Running METplus

This use case can be run by passing in the conf file to the run script:

```
run_metplus.py /path/to/METplus/parm/use_cases/met_tool_wrapper/GFDLTracker/GFDLTracker_TC.
↪conf
```

See the *Running METplus* (page 26) section of the User's Guide for more information on how to run use cases.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in gfdl_tracker/tc (relative to **OUTPUT_BASE**) and will contain the following file:

- hwrf.2016090600.track.txt
- input.201609060000.nml

Keywords

Note:

- GFDLTrackerToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.6.3 GFDLTracker: Extra Tropical Cyclone Use Case

met_tool_wrapper/GFDLTracker/GFDLTracker_ETC.conf

Scientific Objective

Setup and run GFDL Tracker applications to track extra tropical cyclones. See *GFDL Tracker (optional)* (page 34) for more information. A genesis vitals file is read into the tracker. This file contains information on storms that were tracked in the previous 2 runs so that additional data is attributed to the correct storm.

Datasets

Forecast: GFS

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 394) section for more information.

METplus Components

This use case utilizes the METplus GFDLTracker wrapper to generate a command to run the GFDL Tracker Fortran applications.

METplus Workflow

GFDLTracker is the only tool called in this example. It processes the following run time:

Init: 2021-07-13 00Z
Forecast lead: All available leads (0 - 198 hour)

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/GFDLTracker/GFDLTracker_ETC.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GFDLTracker/
↪GFDLTracker_ETC.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GFDLTracker


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
```

(continues on next page)

(continued from previous page)

```
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2021071300
INIT_END = 2021071300
INIT_INCREMENT = 6H

LEAD_SEQ = *


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

GFDL_TRACKER_INPUT_DIR = {INPUT_BASE}/met_test/gfdl/gfs
GFDL_TRACKER_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/gfs.t{init?fmt=%H}z.pgrb2.1p00.f{lead?fmt=
↪%3H}

GFDL_TRACKER_TC_VITALS_INPUT_DIR = {GFDL_TRACKER_INPUT_DIR}
GFDL_TRACKER_TC_VITALS_INPUT_TEMPLATE = syndat_tcvitals.{init?fmt=%Y}

GFDL_TRACKER_GEN_VITALS_INPUT_DIR = {GFDL_TRACKER_INPUT_DIR}
GFDL_TRACKER_GEN_VITALS_INPUT_TEMPLATE = genesis.vitals.gfso.glbl.{init?fmt=%Y%m}

GFDL_TRACKER_OUTPUT_DIR = {OUTPUT_BASE}/gfdl_tracker/etc
GFDL_TRACKER_OUTPUT_TEMPLATE = gfs.{init?fmt=%Y%m%d%H}.etc.txt


###
# GFDLTracker Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gfdltracker
###

GFDL_TRACKER_GRIB_VERSION = 2

GFDL_TRACKER_NML_TEMPLATE_FILE = {PARM_BASE}/use_cases/met_tool_wrapper/GFDLTracker/template.
↪nml

GFDL_TRACKER_DATEIN_INP_MODEL = 1
GFDL_TRACKER_DATEIN_INP_MODTYP = "global"
GFDL_TRACKER_DATEIN_INP_LT_UNITS = "hours"
GFDL_TRACKER_DATEIN_INP_FILE_SEQ = "multi"
GFDL_TRACKER_DATEIN_INP_NESTTYP = "fixed"
```

(continues on next page)

(continued from previous page)

```
GFDL_TRACKER_ATCFINFO_ATCFNUM = 81
GFDL_TRACKER_ATCFINFO_ATCFNAME = "GFML"
GFDL_TRACKER_ATCFINFO_ATCFFREQ = 600

GFDL_TRACKER_TRACKERINFO_TYPE = "midlat"
GFDL_TRACKER_TRACKERINFO_MSLPTHRESH = 0.0015
GFDL_TRACKER_TRACKERINFO_USE_BACKUP_MSLP_GRAD_CHECK = True
GFDL_TRACKER_TRACKERINFO_V850THRESH = 1.5
GFDL_TRACKER_TRACKERINFO_USE_BACKUP_850_VT_CHECK = True
GFDL_TRACKER_TRACKERINFO_ENABLE_TIMING = 1
GFDL_TRACKER_TRACKERINFO_GRIDTYPE = "global"
GFDL_TRACKER_TRACKERINFO_CONTINT = 100.0
GFDL_TRACKER_TRACKERINFO_WANT_OCI = T
GFDL_TRACKER_TRACKERINFO_OUT_VIT = True
GFDL_TRACKER_TRACKERINFO_USE_LAND_MASK = False
GFDL_TRACKER_TRACKERINFO_INP_DATA_TYPE = "grib"
GFDL_TRACKER_TRACKERINFO_GRIBVER = 2
GFDL_TRACKER_TRACKERINFO_G2_JPDTN = 0
GFDL_TRACKER_TRACKERINFO_G2_MSLP_PARM_ID = 1
GFDL_TRACKER_TRACKERINFO_G1_MSLP_PARM_ID = 2
GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_TYP = 105
GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_VAL = 10

GFDL_TRACKER_TRACKERINFO_WESTBD = 0
GFDL_TRACKER_TRACKERINFO_EASTBD = 358
GFDL_TRACKER_TRACKERINFO_SOUTHBD = -89
GFDL_TRACKER_TRACKERINFO_NORTHBD = 89

GFDL_TRACKER_PHASEINFO_PHASEFLAG = True
GFDL_TRACKER_PHASEINFO_PHASESCHEME = "both"
GFDL_TRACKER_PHASEINFO_WCORE_DEPTH = 1.0

GFDL_TRACKER_STRUCTINFO_STRUCTFLAG = False
GFDL_TRACKER_STRUCTINFO_IKEFLAG = False

GFDL_TRACKER_FNAMEINFO_GMODNAME = "gfs"
GFDL_TRACKER_FNAMEINFO_RUNDESCR = "t{init?fmt=%H}z.pgrb2"
GFDL_TRACKER_FNAMEINFO_ATCFDESCR = "1p00"

GFDL_TRACKER_WAITINFO_USE_WAITFOR = True
GFDL_TRACKER_WAITINFO_WAIT_MIN_AGE = 10
GFDL_TRACKER_WAITINFO_WAIT_MIN_SIZE = 100
GFDL_TRACKER_WAITINFO_WAIT_MAX_WAIT = 3600
GFDL_TRACKER_WAITINFO_WAIT_SLEEPTIME = 5
GFDL_TRACKER_WAITINFO_USE_PER_FCST_COMMAND = True
```

(continues on next page)

(continued from previous page)

```
GFDL_TRACKER_WAITINFO_PER_FCST_COMMAND = "./deliver %[FHOUR] %[FMIN]"

GFDL_TRACKER_NETCDFINFO_LAT_NAME = ""
GFDL_TRACKER_NETCDFINFO_LMASKNAME = ""
GFDL_TRACKER_NETCDFINFO_LON_NAME = ""
GFDL_TRACKER_NETCDFINFO_MSLPNAME = ""
GFDL_TRACKER_NETCDFINFO_NETCDF_FILENAME = ""
GFDL_TRACKER_NETCDFINFO_NUM_NETCDF_VARS = 0
GFDL_TRACKER_NETCDFINFO_RV700NAME = ""
GFDL_TRACKER_NETCDFINFO_RV850NAME = ""
GFDL_TRACKER_NETCDFINFO_TIME_NAME = ""
GFDL_TRACKER_NETCDFINFO_TIME_UNITS = ""
GFDL_TRACKER_NETCDFINFO_TMEAN_300_500_NAME = ""
GFDL_TRACKER_NETCDFINFO_U500NAME = ""
GFDL_TRACKER_NETCDFINFO_U700NAME = ""
GFDL_TRACKER_NETCDFINFO_U850NAME = ""
GFDL_TRACKER_NETCDFINFO_USFCNAME = ""
GFDL_TRACKER_NETCDFINFO_V500NAME = ""
GFDL_TRACKER_NETCDFINFO_V700NAME = ""
GFDL_TRACKER_NETCDFINFO_V850NAME = ""
GFDL_TRACKER_NETCDFINFO_VSFCNAME = ""
GFDL_TRACKER_NETCDFINFO_Z200NAME = ""
GFDL_TRACKER_NETCDFINFO_Z300NAME = ""
GFDL_TRACKER_NETCDFINFO_Z350NAME = ""
GFDL_TRACKER_NETCDFINFO_Z400NAME = ""
GFDL_TRACKER_NETCDFINFO_Z450NAME = ""
GFDL_TRACKER_NETCDFINFO_Z500NAME = ""
GFDL_TRACKER_NETCDFINFO_Z550NAME = ""
GFDL_TRACKER_NETCDFINFO_Z600NAME = ""
GFDL_TRACKER_NETCDFINFO_Z650NAME = ""
GFDL_TRACKER_NETCDFINFO_Z700NAME = ""
GFDL_TRACKER_NETCDFINFO_Z750NAME = ""
GFDL_TRACKER_NETCDFINFO_Z800NAME = ""
GFDL_TRACKER_NETCDFINFO_Z850NAME = ""
GFDL_TRACKER_NETCDFINFO_Z900NAME = ""

GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA850 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA700 = False
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC850 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC700 = False
GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH850 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH700 = False
GFDL_TRACKER_USER_WANTS_TO_TRACK_MSLP = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRCSFC = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETASFC = True
```

(continues on next page)

(continued from previous page)

```
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK500850 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200500 = True
GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200850 = True

GFDL_TRACKER_VERBOSE_VERB = 3
GFDL_TRACKER_VERBOSE_VERB_G2 = 0
```

GFDL Tracker Configuration

METplus replaces values in the template configuration files read by the tracker based on user settings in the METplus configuration file.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

```
&datein
  inp%bcc = ${METPLUS_DATEIN_INP_BCC},
  inp%byy = ${METPLUS_DATEIN_INP_BYY},
  inp%bmm = ${METPLUS_DATEIN_INP_BMM},
  inp%bdd = ${METPLUS_DATEIN_INP_BDD},
  inp%bhh = ${METPLUS_DATEIN_INP_BHH},
  inp%model = ${METPLUS_DATEIN_INP_MODEL},
  inp%modtyp = ${METPLUS_DATEIN_INP_MODTYP},
  inp%lt_units = ${METPLUS_DATEIN_INP_LT_UNITS},
  inp%file_seq = ${METPLUS_DATEIN_INP_FILE_SEQ},
  inp%nesttyp = ${METPLUS_DATEIN_INP_NESTTYP},
/

&atcfinfo
  atcfnum = ${METPLUS_ATCFINFO_ATCFNUM},
  atcfname = ${METPLUS_ATCFINFO_ATCFNAME},
  atcfymdh = ${METPLUS_ATCFINFO_ATCFYMDH},
  atcffreq = ${METPLUS_ATCFINFO_ATCFFREQ},
/

&trackerinfo
  trkrinfo%type = ${METPLUS_TRACKERINFO_TYPE},
  trkrinfo%mslpthresh = ${METPLUS_TRACKERINFO_MSLPTHRESH},
  trkrinfo%use_backup_mslp_grad_check = ${METPLUS_TRACKERINFO_USE_BACKUP_MSLP_GRAD_CHECK},
  trkrinfo%v850thresh = ${METPLUS_TRACKERINFO_V850THRESH},
  trkrinfo%use_backup_850_vt_check = ${METPLUS_TRACKERINFO_USE_BACKUP_850_VT_CHECK},
  trkrinfo%enable_timing = ${METPLUS_TRACKERINFO_ENABLE_TIMING},
  trkrinfo%gridtype = ${METPLUS_TRACKERINFO_GRIDTYPE},
  trkrinfo%contint = ${METPLUS_TRACKERINFO_CONTINT},
  trkrinfo%want_oci = ${METPLUS_TRACKERINFO_WANT_OCI},
```

(continues on next page)

(continued from previous page)

```
  trkrinfo%out_vit = ${METPLUS_TRACKERINFO_OUT_VIT},
  trkrinfo%use_land_mask = ${METPLUS_TRACKERINFO_USE_LAND_MASK},
  trkrinfo%inp_data_type = ${METPLUS_TRACKERINFO_INP_DATA_TYPE},
  trkrinfo%gribver = ${METPLUS_TRACKERINFO_GRIBVER},
  trkrinfo%g2_jpdtn = ${METPLUS_TRACKERINFO_G2_JPDTN},
  trkrinfo%g2_mslp_parm_id = ${METPLUS_TRACKERINFO_G2_MSLP_PARM_ID},
  trkrinfo%g1_mslp_parm_id = ${METPLUS_TRACKERINFO_G1_MSLP_PARM_ID},
  trkrinfo%g1_sfcwind_lev_typ = ${METPLUS_TRACKERINFO_G1_SFCWIND_LEV_TYP},
  trkrinfo%g1_sfcwind_lev_val = ${METPLUS_TRACKERINFO_G1_SFCWIND_LEV_VAL},
  trkrinfo%westbd = ${METPLUS_TRACKERINFO_WESTBD},
  trkrinfo%eastbd = ${METPLUS_TRACKERINFO_EASTBD},
  trkrinfo%southbd = ${METPLUS_TRACKERINFO_SOUTHBD},
  trkrinfo%northbd = ${METPLUS_TRACKERINFO_NORTHBD},
/

&phaseinfo
  phaseflag = ${METPLUS_PHASEINFO_PHASEFLAG},
  phasescheme = ${METPLUS_PHASEINFO_PHASESCHEME},
  wcore_depth = ${METPLUS_PHASEINFO_WCORE_DEPTH},
/

&structinfo
  structflag = ${METPLUS_STRUCTINFO_STRUCTFLAG},
  ikeflag = ${METPLUS_STRUCTINFO_IKEFLAG},
/

&fnameinfo
  gmodname = ${METPLUS_FNAMEINFO_GMODNAME},
  rundescr = ${METPLUS_FNAMEINFO_RUNDESCR},
  atcfdescr = ${METPLUS_FNAMEINFO_ATCFDESCR},
/

&waitinfo
  use_waitfor = ${METPLUS_WAITINFO_USE_WAITFOR},
  wait_min_age = ${METPLUS_WAITINFO_WAIT_MIN_AGE},
  wait_min_size = ${METPLUS_WAITINFO_WAIT_MIN_SIZE},
  wait_max_wait = ${METPLUS_WAITINFO_WAIT_MAX_WAIT},
  wait_sleeptime = ${METPLUS_WAITINFO_WAIT_SLEEPTIME},
  use_per_fcst_command = ${METPLUS_WAITINFO_USE_PER_FCST_COMMAND},
  per_fcst_command = ${METPLUS_WAITINFO_PER_FCST_COMMAND},
/

&netcdflist
  netcdfinfo%lat_name = ${METPLUS_NETCDFINFO_LAT_NAME},
  netcdfinfo%lmaskname = ${METPLUS_NETCDFINFO_LMASKNAME},
```

(continues on next page)

(continued from previous page)

```
  netcdfinfo%lon_name = ${METPLUS_NETCDFINFO_LON_NAME},
  netcdfinfo%mslpname = ${METPLUS_NETCDFINFO_MSLPNAME},
  netcdfinfo%netcdf_filename = ${METPLUS_NETCDFINFO_NETCDF_FILENAME},
  netcdfinfo%num_netcdf_vars = ${METPLUS_NETCDFINFO_NUM_NETCDF_VARS},
  netcdfinfo%rv700name = ${METPLUS_NETCDFINFO_RV700NAME},
  netcdfinfo%rv850name = ${METPLUS_NETCDFINFO_RV850NAME},
  netcdfinfo%time_name = ${METPLUS_NETCDFINFO_TIME_NAME},
  netcdfinfo%time_units = ${METPLUS_NETCDFINFO_TIME_UNITS},
  netcdfinfo%tmean_300_500_name = ${METPLUS_NETCDFINFO_TMEAN_300_500_NAME},
  netcdfinfo%u500name = ${METPLUS_NETCDFINFO_U500NAME},
  netcdfinfo%u700name = ${METPLUS_NETCDFINFO_U700NAME},
  netcdfinfo%u850name = ${METPLUS_NETCDFINFO_U850NAME},
  netcdfinfo%usfcname = ${METPLUS_NETCDFINFO_USFCNAME},
  netcdfinfo%v500name = ${METPLUS_NETCDFINFO_V500NAME},
  netcdfinfo%v700name = ${METPLUS_NETCDFINFO_V700NAME},
  netcdfinfo%v850name = ${METPLUS_NETCDFINFO_V850NAME},
  netcdfinfo%vsfcname = ${METPLUS_NETCDFINFO_VSFCNAME},
  netcdfinfo%z200name = ${METPLUS_NETCDFINFO_Z200NAME},
  netcdfinfo%z300name = ${METPLUS_NETCDFINFO_Z300NAME},
  netcdfinfo%z350name = ${METPLUS_NETCDFINFO_Z350NAME},
  netcdfinfo%z400name = ${METPLUS_NETCDFINFO_Z400NAME},
  netcdfinfo%z450name = ${METPLUS_NETCDFINFO_Z450NAME},
  netcdfinfo%z500name = ${METPLUS_NETCDFINFO_Z500NAME},
  netcdfinfo%z550name = ${METPLUS_NETCDFINFO_Z550NAME},
  netcdfinfo%z600name = ${METPLUS_NETCDFINFO_Z600NAME},
  netcdfinfo%z650name = ${METPLUS_NETCDFINFO_Z650NAME},
  netcdfinfo%z700name = ${METPLUS_NETCDFINFO_Z700NAME},
  netcdfinfo%z750name = ${METPLUS_NETCDFINFO_Z750NAME},
  netcdfinfo%z800name = ${METPLUS_NETCDFINFO_Z800NAME},
  netcdfinfo%z850name = ${METPLUS_NETCDFINFO_Z850NAME},
  netcdfinfo%z900name = ${METPLUS_NETCDFINFO_Z900NAME},
/

&parmpreflist
  user_wants_to_track_zeta700 = ${METPLUS_USER_WANTS_TO_TRACK_ZETA700},
  user_wants_to_track_wcirc850 = ${METPLUS_USER_WANTS_TO_TRACK_WCIRC850},
  user_wants_to_track_wcirc700 = ${METPLUS_USER_WANTS_TO_TRACK_WCIRC700},
  user_wants_to_track_gph850 = ${METPLUS_USER_WANTS_TO_TRACK_GPH850},
  user_wants_to_track_gph700 = ${METPLUS_USER_WANTS_TO_TRACK_GPH700},
  user_wants_to_track_mslp = ${METPLUS_USER_WANTS_TO_TRACK_MSLP},
  user_wants_to_track_wcircsfc = ${METPLUS_USER_WANTS_TO_TRACK_WCIRCSFC},
  user_wants_to_track_zetasfc = ${METPLUS_USER_WANTS_TO_TRACK_ZETASFC},
  user_wants_to_track_thick500850 = ${METPLUS_USER_WANTS_TO_TRACK_THICK500850},
  user_wants_to_track_thick200500 = ${METPLUS_USER_WANTS_TO_TRACK_THICK200500},
  user_wants_to_track_thick200850 = ${METPLUS_USER_WANTS_TO_TRACK_THICK200850},
```

(continues on next page)

(continued from previous page)

```
  user_wants_to_track_zeta850 = ${METPLUS_USER_WANTS_TO_TRACK_ZETA850},
/

&verbose
  verb = ${METPLUS_VERBOSE_VERB},
  verb_g2 = ${METPLUS_VERBOSE_VERB_G2},
/
```

Running METplus

This use case can be run by passing in the conf file to the run script:

```
run_metplus.py /path/to/METplus/parm/use_cases/met_tool_wrapper/GFDLTracker/GFDLTracker_ETC.
↪conf
```

See the *Running METplus* (page 26) section of the User's Guide for more information on how to run use cases.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in gfdl_tracker/etc (relative to **OUTPUT_BASE**) and will contain the following file:

- gfs.2021071300.etc.txt
- input.202107130000.nml

Keywords

Note:

- GFDLTrackerToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.7 GempakToCF

7.1.32.7.1 GempakToCF: Basic Use Case

met_tool_wrapper/GempakToCF/GempakToCF.conf

Scientific Objective

None. Simply converting data to a format that MET can read.

Datasets

Observations: MRMS QPE

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 397) section for more information.

Data Source: Unknown

External Dependencies

GempakToCF.jar

GempakToCF is an external tool that utilizes the Unidata NetCDF-Java package. The jar file that can be used to run the utility is available here: https://dtcenter.org/sites/default/files/community-code/metplus/utilities/GempakToCF.jar

See the METplus Configuration section below for information on how to configure METplus to find the jar file.

More information on the package used to create the file is here: https://www.unidata.ucar.edu/software/netcdf-java

METplus Components

This use case utilizes the METplus GempakToCF wrapper to generate a command to run GempakToCF (external) if all required files are found.

METplus Workflow

GempakToCF is the only tool called in this example. It processes the following run times:

Init: 2017-06-22 0Z

Init: 2017-06-22 12Z

METplus Configuration

To enable Gempak support, you must set [exe] *GEMPAKTOCF_JAR* in your user METplus configuration file.:

[exe] *GEMPAKTOCF_JAR* = /path/to/GempakToCF.jar

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/GempakToCF/GempakToCF.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GempakToCF/GempakToCF.
↪html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GempakToCF


###
# Time Info
```

(continues on next page)

(continued from previous page)

```
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG=2017062200
VALID_END=2017062212
VALID_INCREMENT=12H

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

GEMPAKTOCF_INPUT_DIR = {INPUT_BASE}/met_test/new/gempak
GEMPAKTOCF_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/mrms_qpe_{valid?fmt=%Y%m%d%H}.grd

GEMPAKTOCF_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/GempakToCF
GEMPAKTOCF_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/mrms_qpe_{valid?fmt=%Y%m%d%H}.nc

GEMPAKTOCF_SKIP_IF_OUTPUT_EXISTS = False
```

Running METplus

This use case can be run two ways:

1) Passing in GempakToCF.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GempakToCF/
↪GempakToCF.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GempakToCF.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GempakToCF/
↪GempakToCF.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/GempakToCF (relative to **OUTPUT_BASE**) and will contain the following file:

- 20170622/mrms_qpe_2017062200.nc
- 20170622/mrms_qpe_2017062212.nc

Keywords

Note:

- GempakToCFToolUseCase
- GEMPAKFileUseCase
- NOAAHMTOrgUseCase
- NOAAWPCOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-GempakToCF.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.8 GenEnsProd

7.1.32.8.1 GenEnsProd: Basic Use Case

met_tool_wrapper/GenEnsProd/GenEnsProd.conf

Scientific Objective

Generate ensemble products. This use case demonstrates how to configure the gen_ens_prod tool if you expect that there will occasionally be missing ensembles. 7 ensemble paths are specified but only 6 of them exist in the sample input data set. The wrapper will mark ensembles that are not found with the MISSING keyword in the file-list file that is read by the tool. Also, one of the ensembles is listed as the control member. The gen_ens_prod application will error and exit if the control member is included in the ensemble list, but the GenEnsProd wrapper will automatically remove the control member from the ensemble list. This makes it easier to configure the tool to change the control member without having to change the ensemble list. The number of expected members (defined with GEN_ENS_PROD_N_MEMBERS) is 6 (7 members - 1 control member). The actual number of ensemble members that will be found in this example is 5 (arw-tom-gep4 is not included). The ens.ens_thresh value (defined by GEN_ENS_PROD_ENS_THRESH) is set to 0.8. There are ~0.833 (5/6) valid ensemble members so the application will run.

Datasets

Input: WRF ARW ensemble 24 hour precipitation accumulation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See the *Running METplus* (page 407) section for more information.

METplus Components

This use case utilizes the METplus GenEnsProd wrapper to generate a command to run the MET tool gen_ens_prod if all required files are found.

METplus Workflow

GenEnsProd is the only tool called in this example. It processes the following run time(s):

Initialization: 2009-12-31 12Z
Forecast Lead: 24 hour

METplus Configuration

parm/use_cases/met_tool_wrapper/GenEnsProd/GenEnsProd.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GenEnsProd/GenEnsProd.
↪html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GenEnsProd


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2009123112
```

(continues on next page)

(continued from previous page)

```
INIT_END=2009123112
INIT_INCREMENT = 12H

LEAD_SEQ = 24H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

GEN_ENS_PROD_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
GEN_ENS_PROD_INPUT_TEMPLATE =
  {init?fmt=%Y%m%d%H}/arw-fer-gep1/d01_{init?fmt=%Y%m%d%H}_{lead?fmt=%3H}00.grib,
  {init?fmt=%Y%m%d%H}/arw-sch-gep2/d01_{init?fmt=%Y%m%d%H}_{lead?fmt=%3H}00.grib,
  {init?fmt=%Y%m%d%H}/arw-tom-gep3/d01_{init?fmt=%Y%m%d%H}_{lead?fmt=%3H}00.grib,
  {init?fmt=%Y%m%d%H}/arw-tom-gep4/d01_{init?fmt=%Y%m%d%H}_{lead?fmt=%3H}00.grib,
  {init?fmt=%Y%m%d%H}/arw-fer-gep5/d01_{init?fmt=%Y%m%d%H}_{lead?fmt=%3H}00.grib,
  {init?fmt=%Y%m%d%H}/arw-sch-gep6/d01_{init?fmt=%Y%m%d%H}_{lead?fmt=%3H}00.grib,
  {init?fmt=%Y%m%d%H}/arw-tom-gep7/d01_{init?fmt=%Y%m%d%H}_{lead?fmt=%3H}00.grib

GEN_ENS_PROD_CTRL_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
GEN_ENS_PROD_CTRL_INPUT_TEMPLATE =
  {init?fmt=%Y%m%d%H}/arw-fer-gep1/d01_{init?fmt=%Y%m%d%H}_{lead?fmt=%3H}00.grib

GEN_ENS_PROD_OUTPUT_DIR = {OUTPUT_BASE}/gen_ens_prod
GEN_ENS_PROD_OUTPUT_TEMPLATE = gen_ens_prod_{valid?fmt=%Y%m%d_%H%M%S}V_ens.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

ENS_VAR1_NAME = APCP
ENS_VAR1_LEVELS = A24
ENS_VAR1_THRESH = >0.0, >=10.0
ENS_VAR1_OPTIONS = ensemble_flag = TRUE

ENS_VAR2_NAME = REFC
ENS_VAR2_LEVELS = L0
ENS_VAR2_THRESH = >=35.0
ENS_VAR2_OPTIONS = GRIB1_ptv = 129
```

(continues on next page)

(continued from previous page)

```
ENS_VAR3_NAME = UGRD
ENS_VAR3_LEVELS = Z10
ENS_VAR3_THRESH = >=5.0

ENS_VAR4_NAME = VGRD
ENS_VAR4_LEVELS = Z10
ENS_VAR4_THRESH = >=5.0

ENS_VAR5_NAME = WIND
ENS_VAR5_LEVELS = Z10
ENS_VAR5_THRESH = >=5.0


###
# GenEnsProd Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#genensprod
###

GEN_ENS_PROD_N_MEMBERS = 6

#LOG_GEN_ENS_PROD_VERBOSITY = 2

# MODEL = WRF
# GEN_ENS_PROD_DESC = NA

#GEN_ENS_PROD_REGRID_TO_GRID = NONE
#GEN_ENS_PROD_REGRID_METHOD = NEAREST
#GEN_ENS_PROD_REGRID_WIDTH = 1
#GEN_ENS_PROD_REGRID_VLD_THRESH = 0.5
#GEN_ENS_PROD_REGRID_SHAPE = SQUARE
#GEN_ENS_PROD_REGRID_CONVERT =
#GEN_ENS_PROD_REGRID_CENSOR_THRESH =
#GEN_ENS_PROD_REGRID_CENSOR_VAL =

#GEN_ENS_PROD_CENSOR_THRESH =
#GEN_ENS_PROD_CENSOR_VAL =
#GEN_ENS_PROD_NORMALIZE =
#GEN_ENS_PROD_CAT_THRESH =
#GEN_ENS_PROD_NC_VAR_STR =

GEN_ENS_PROD_ENS_THRESH = 0.8
#GEN_ENS_PROD_VLD_THRESH = 1.0

#GEN_ENS_PROD_NBRHD_PROB_WIDTH = 5
#GEN_ENS_PROD_NBRHD_PROB_SHAPE = CIRCLE
```

(continues on next page)

(continued from previous page)

```
#GEN_ENS_PROD_NBRHD_PROB_VLD_THRESH = 0.0

#GEN_ENS_PROD_NMEP_SMOOTH_VLD_THRESH = 0.0
#GEN_ENS_PROD_NMEP_SMOOTH_SHAPE = CIRCLE
#GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_DX = 81.27
#GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_RADIUS = 120
#GEN_ENS_PROD_NMEP_SMOOTH_METHOD = GAUSSIAN
#GEN_ENS_PROD_NMEP_SMOOTH_WIDTH = 1

#GEN_ENS_PROD_CLIMO_MEAN_FILE_NAME =
#GEN_ENS_PROD_CLIMO_MEAN_FIELD =
#GEN_ENS_PROD_CLIMO_MEAN_REGRID_METHOD =
#GEN_ENS_PROD_CLIMO_MEAN_REGRID_WIDTH =
#GEN_ENS_PROD_CLIMO_MEAN_REGRID_VLD_THRESH =
#GEN_ENS_PROD_CLIMO_MEAN_REGRID_SHAPE =
#GEN_ENS_PROD_CLIMO_MEAN_TIME_INTERP_METHOD =
#GEN_ENS_PROD_CLIMO_MEAN_MATCH_MONTH =
#GEN_ENS_PROD_CLIMO_MEAN_DAY_INTERVAL = 31
#GEN_ENS_PROD_CLIMO_MEAN_HOUR_INTERVAL = 6

#GEN_ENS_PROD_CLIMO_STDEV_FILE_NAME =
#GEN_ENS_PROD_CLIMO_STDEV_FIELD =
#GEN_ENS_PROD_CLIMO_STDEV_REGRID_METHOD =
#GEN_ENS_PROD_CLIMO_STDEV_REGRID_WIDTH =
#GEN_ENS_PROD_CLIMO_STDEV_REGRID_VLD_THRESH =
#GEN_ENS_PROD_CLIMO_STDEV_REGRID_SHAPE =
#GEN_ENS_PROD_CLIMO_STDEV_TIME_INTERP_METHOD =
#GEN_ENS_PROD_CLIMO_STDEV_MATCH_MONTH =
#GEN_ENS_PROD_CLIMO_STDEV_DAY_INTERVAL = 31
#GEN_ENS_PROD_CLIMO_STDEV_HOUR_INTERVAL = 6

#GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON = TRUE
#GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN = TRUE
#GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV = TRUE
#GEN_ENS_PROD_ENSEMBLE_FLAG_MINUS = TRUE
#GEN_ENS_PROD_ENSEMBLE_FLAG_PLUS = TRUE
#GEN_ENS_PROD_ENSEMBLE_FLAG_MIN = TRUE
#GEN_ENS_PROD_ENSEMBLE_FLAG_MAX = TRUE
#GEN_ENS_PROD_ENSEMBLE_FLAG_RANGE = TRUE
#GEN_ENS_PROD_ENSEMBLE_FLAG_VLD_COUNT = TRUE
#GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY = TRUE
#GEN_ENS_PROD_ENSEMBLE_FLAG_NEP = FALSE
#GEN_ENS_PROD_ENSEMBLE_FLAG_NMEP = FALSE
#GEN_ENS_PROD_ENSEMBLE_FLAG_CLIMO = FALSE
#GEN_ENS_PROD_ENSEMBLE_FLAG_CLIMO_CDP = FALSE
```

(continues on next page)

(continued from previous page)

```
#GEN_ENS_PROD_ENS_MEMBER_IDS =
#GEN_ENS_PROD_CONTROL_ID =
```

MET Configuration

Note: See the *GenEnsProd MET Configuration* (page 121) section of the User's Guide for more information on the environment variables used in the file below.

---

parm/met_config/GenEnsProdConfig_wrapped

```
////////////////////////////////////////////////////////////////////////////////
//
// Gen-Ens-Prod configuration file.
//
// For additional information, please see the MET Users Guide.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
//desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// May be set separately in each "field" entry
//
//censor_thresh =
${METPLUS_CENSOR_THRESH}

//censor_val    =
${METPLUS_CENSOR_VAL}

//normalize =
${METPLUS_NORMALIZE}

//cat_thresh    =
${METPLUS_CAT_THRESH}

//nc_var_str    =
${METPLUS_NC_VAR_STR}

//
// Ensemble fields to be processed
//
ens = {
   //file_type =
   ${METPLUS_ENS_FILE_TYPE}

   //ens_thresh =
   ${METPLUS_ENS_THRESH}

   //vld_thresh =
   ${METPLUS_VLD_THRESH}

   //field =
   ${METPLUS_ENS_FIELD}

}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}


////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Neighborhood ensemble probabilities
//
//nbrhd_prob = {
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
//nmep_smooth = {
${METPLUS_NMEP_SMOOTH_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}

//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Ensemble product output types
// May be set separately in each "ens.field" entry
//
//ensemble_flag = {
${METPLUS_ENSEMBLE_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//version = "V10.1.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

Provide the use case .conf configuration file to the run_metplus.py script.

/path/to/METplus/parm/use_cases/met_tool_wrapper/GenEnsProd/GenEnsProd.conf

See the *Running METplus* (page 26) section of the System Configuration chapter for more details.

Expected Output

A successful run will output the following to the screen and the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/gen_ens_prod (relative to **OUTPUT_BASE**) and will contain the following file(s):

- gen_ens_prod_20100101_120000V_ens.nc

A file-list file will also be generated in stage/file_lists called:

- 20091231120000_24_gen_ens_prod.txt

It should contain a list of 6 files in {INPUT_BASE} with 1 file marked as missing because it was not found:

```
file_list
{INPUT_BASE}/met_test/data/sample_fcst/2009123112/arw-sch-gep2/d01_2009123112_02400.grib
{INPUT_BASE}/met_test/data/sample_fcst/2009123112/arw-tom-gep3/d01_2009123112_02400.grib
MISSING/{INPUT_BASE}/met_test/data/sample_fcst/2009123112/arw-tom-gep4/d01_2009123112_02400.
↪grib
{INPUT_BASE}/met_test/data/sample_fcst/2009123112/arw-fer-gep5/d01_2009123112_02400.grib
{INPUT_BASE}/met_test/data/sample_fcst/2009123112/arw-sch-gep6/d01_2009123112_02400.grib
{INPUT_BASE}/met_test/data/sample_fcst/2009123112/arw-tom-gep7/d01_2009123112_02400.grib
```

Keywords

Note:

- GenEnsProdToolUseCase
- GRIBFileUseCase
- EnsembleAppUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1997) to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-GenEnsProd.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.9 GenVxMask

7.1.32.9.1 GenVxMask: Basic Use Case

met_tool_wrapper/GenVxMask/GenVxMask.conf

Scientific Objective

Creating masking region files to be used by other MET tools.

Datasets

Input Grid: GFS

Mask: CONUS polyline file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 410) section for more information.

METplus Components

This use case utilizes the METplus GenVxMask wrapper to generate a command to run the MET tool GenVxMask if all required files are found.

METplus Workflow

GenVxMask is the only tool called in this example. It processes the following run time:

Initialization: 2012-04-09_0Z
Forecast Lead: 12 hour

The input file is read to define the output grid and the CONUS polyline file is applied to create the mask.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/GenVxMask/GenVxMask.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GenVxMask/GenVxMask.
↪html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GenVxMask


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2012040900
INIT_END = 2012040900
INIT_INCREMENT = 1M

LEAD_SEQ = 12H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

GEN_VX_MASK_INPUT_DIR =
GEN_VX_MASK_INPUT_TEMPLATE = {INPUT_BASE}/met_test/new/gfs/gfs_{init?fmt=%Y%m%d%H}_F{lead?
↪fmt=%3H}.grib

GEN_VX_MASK_INPUT_MASK_DIR =
GEN_VX_MASK_INPUT_MASK_TEMPLATE = {INPUT_BASE}/met_test/data/poly/CONUS.poly

GEN_VX_MASK_OUTPUT_DIR =
GEN_VX_MASK_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/GenVxMask/POLY_GFS_LATLON_CONUS_
↪mask.nc


GEN_VX_MASK_SKIP_IF_OUTPUT_EXISTS = False

GEN_VX_MASK_FILE_WINDOW_BEGIN = 0
GEN_VX_MASK_FILE_WINDOW_END = 0


###
# GenVxMask Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#genvxmask
###

#LOG_GEN_VX_MASK_VERBOSITY = 2

GEN_VX_MASK_OPTIONS = -type poly
```

MET Configuration

None. GenVxMask does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in the use case config file then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GenVxMask/GenVxMask.
↪conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in the use case config file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GenVxMask/GenVxMask.
↪conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/GenVxMask (relative to **OUTPUT_BASE**) and will contain the following file:

- POLY_GFS_LATLON_CONUS_mask.nc

Keywords

Note:

- GenVxMaskToolUseCase
- GRIBFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-GenVxMask.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.9.2 GenVxMask: Multiple Masks

met_tool_wrapper/GenVxMask/GenVxMask_multiple.conf

Scientific Objective

Creating masking region files to be used by other MET tools. This use case applies multiple masks (latitude restriction, then longitude restriction) to the input grid.

Datasets

Input Grid: WRF

Masks: Latitude bounds, longitude bounds

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 414) section for more information.

METplus Components

This use case utilizes the METplus GenVxMask wrapper to generate a command to run the MET tool GenVxMask if all required files are found.

METplus Workflow

GenVxMask is the only tool called in this example. It processes the following run time:

Initialization: 2005-08-07 0Z
Forecast Lead: 24 hour

The input file is read to define the output grid. First the latitude bounds specified with the -thresh argument are applied to the input file, creating a temporary intermediate file. Then a longitude threshold is applied to the temporary file, creating the final output file.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/GenVxMask/GenVxMask_multiple.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GenVxMask/GenVxMask_
↪multiple.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GenVxMask


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2005080700
INIT_END = 2005080700
INIT_INCREMENT = 1M

LEAD_SEQ = 24H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

GEN_VX_MASK_INPUT_DIR =
GEN_VX_MASK_INPUT_TEMPLATE = {INPUT_BASE}/met_test/data/sample_fcst/{init?fmt=%Y%m%d%H}/
↪wrfprs_ruc13_{lead?fmt=%2H}.tm00_G212

GEN_VX_MASK_INPUT_MASK_DIR =
GEN_VX_MASK_INPUT_MASK_TEMPLATE = LATLON_GRID, LATLON_GRID

GEN_VX_MASK_OUTPUT_DIR =
GEN_VX_MASK_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/GenVxMask/LAT_LON_mask.nc

GEN_VX_MASK_FILE_WINDOW_BEGIN = 0
GEN_VX_MASK_FILE_WINDOW_END = 0

GEN_VX_MASK_SKIP_IF_OUTPUT_EXISTS = False


###
# GenVxMask Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#genvxmask
###

#LOG_GEN_VX_MASK_VERBOSITY = 2

GEN_VX_MASK_OPTIONS = -type lat -thresh 'ge30&&le50', -type lon -thresh 'le-70&&ge-130' -
↪intersection -name lat_lon_mask
```

MET Configuration

None. GenVxMask does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in the use case config file then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GenVxMask/GenVxMask_
↪multiple.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in the use case config file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GenVxMask/GenVxMask_
↪multiple.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/GenVxMask (relative to **OUTPUT_BASE**) and will contain the following file:

- LAT_LON_mask.nc

Keywords

Note:

- GenVxMaskToolUseCase
- GRIBFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-GenVxMask.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.9.3 GenVxMask: Using Arguments

met_tool_wrapper/GenVxMask/GenVxMask_with_arguments.conf

Scientific Objective

Creating masking region files to be used by other MET tools. This use case adds command line arguments to define the mask applied to the input grid.

Datasets

Input Grid: WRF Precipitation

Mask: WRF Temperature

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 418) section for more information.

METplus Components

This use case utilizes the METplus GenVxMask wrapper to generate a command to run the MET tool GenVxMask if all required files are found.

METplus Workflow

GenVxMask is the only tool called in this example. It processes the following run time:

Initialization: 2005-08-07 0Z
Forecast Lead: 24 hour

The input file is read to define the output grid. Command line arguments are added to the call to define which data to use to apply a mask.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/GenVxMask/GenVxMask_with_arguments.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GenVxMask/GenVxMask_
↪with_arguments.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GenVxMask


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2005080700
INIT_END = 2005080700
INIT_INCREMENT = 1M

LEAD_SEQ = 24H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

GEN_VX_MASK_INPUT_DIR =
GEN_VX_MASK_INPUT_TEMPLATE = {INPUT_BASE}/met_test/data/sample_fcst/{init?fmt=%Y%m%d%H}/
↪wrfprs_ruc13_{lead?fmt=%2H}.tm00_G212

GEN_VX_MASK_INPUT_MASK_DIR =
GEN_VX_MASK_INPUT_MASK_TEMPLATE = {GEN_VX_MASK_INPUT_TEMPLATE}

GEN_VX_MASK_OUTPUT_DIR =
GEN_VX_MASK_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/GenVxMask/DATA_INPUT_FIELD_APCP_
↪{lead?fmt=%2H}_where_TMP_Z2_le300.nc

GEN_VX_MASK_SKIP_IF_OUTPUT_EXISTS = False

GEN_VX_MASK_FILE_WINDOW_BEGIN = 0
GEN_VX_MASK_FILE_WINDOW_END = 0


###
# GenVxMask Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#genvxmask
###

#LOG_GEN_VX_MASK_VERBOSITY = 2

GEN_VX_MASK_OPTIONS = -type data -input_field 'name="APCP"; level="A{lead?fmt=%2H}";' -mask_
↪field  'name="TMP";  level="Z2";' -thresh 'gt300' -value -9999 -name "APCP_{lead?fmt=%2H}_
↪where_TMP_Z2_le300"
```

MET Configuration

None. GenVxMask does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in the use case config file then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GenVxMask/GenVxMask_
↪with_arguments.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in the use case config file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GenVxMask/GenVxMask_
↪with_arguments.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/GenVxMask (relative to **OUTPUT_BASE**) and will contain the following file:

- DATA_INPUT_FIELD_APCP_24_where_TMP_Z2_le300.nc

Keywords

Note:

- GenVxMaskToolUseCase
- GRIBFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-GenVxMask.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.10 GridDiag

7.1.32.10.1 GridDiag: Basic Use Case

met_tool_wrapper/GridDiag/GridDiag.conf

Scientific Objective

The Grid-Diag tool creates histograms (probability distributions when normalized) for an arbitrary collection of data fields and levels.

Datasets

Data: GFS FV3

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 424) section for more information.

METplus Components

This use case utilizes the METplus GridDiag wrapper to search for files that are valid at a given run time and generate a command to run the MET tool grid_diag if all required files are found.

METplus Workflow

GridDiag is the only tool called in this example. It processes the following run times:

Init: 2016-09-29_0Z
Forecast leads: 141, 144, and 147 hours

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/GridDiag/GridDiag.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GridDiag/GridDiag.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridDiag


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

GRID_DIAG_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2016092900
INIT_END = 2016092900
INIT_INCREMENT = 21600

LEAD_SEQ = 141, 144, 147


###
# File I/O
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

GRID_DIAG_INPUT_DIR = {INPUT_BASE}/met_test/new/model_data/grib2/gfs_fv3
GRID_DIAG_INPUT_TEMPLATE = gfs.subset.t00z.pgrb2.0p25.f{lead?fmt=%H}, gfs.subset.t00z.pgrb2.
↪0p25.f{lead?fmt=%H}

GRID_DIAG_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/GridDiag
GRID_DIAG_OUTPUT_TEMPLATE = grid_diag_out.nc



###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

BOTH_VAR1_NAME = APCP
BOTH_VAR1_LEVELS = L0
BOTH_VAR1_OPTIONS = n_bins = 55; range  = [0, 55];

BOTH_VAR2_NAME = PWAT
BOTH_VAR2_LEVELS = L0
BOTH_VAR2_OPTIONS = n_bins = 35; range  = [35, 70];



###
# GridDiag Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#griddiag
###

#LOG_GRID_DIAG_VERBOSITY = 2

GRID_DIAG_DESC = GFS

GRID_DIAG_CONFIG_FILE = {PARM_BASE}/met_config/GridDiagConfig_wrapped

#GRID_DIAG_REGRID_TO_GRID = NONE
#GRID_DIAG_REGRID_METHOD = NEAREST
#GRID_DIAG_REGRID_WIDTH = 1
#GRID_DIAG_REGRID_VLD_THRESH = 0.5
#GRID_DIAG_REGRID_SHAPE = SQUARE
#GRID_DIAG_REGRID_CONVERT =
#GRID_DIAG_REGRID_CENSOR_THRESH =
#GRID_DIAG_REGRID_CENSOR_VAL =
```

(continues on next page)

(continued from previous page)

```
GRID_DIAG_MASK_POLY = MET_BASE/poly/SAO.poly
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridDiag MET Configuration* (page 145) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Diag configuration file.
//
// For additional information, see the MET_BASE/config/GridDiagConfig_default file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Description
//
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Output grid
//
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
${METPLUS_CENSOR_THRESH}
${METPLUS_CENSOR_VAL}
```

(continues on next page)

(continued from previous page)

```
//
// Data fields
//
${METPLUS_DATA_DICT}

${METPLUS_MASK_DICT}

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridDiag.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridDiag/GridDiag.
↪conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridDiag.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridDiag/GridDiag.
↪conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/GridDiag (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_diag_out.nc

Keywords

Note:

- GridDiagToolUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-GridDiag.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.11 GridStat

7.1.32.11.1 GridStat: Using Python Embedding

met_tool_wrapper/GridStat/GridStat_python_embedding.conf

Scientific Objective

Compare dummy forecast data to dummy observations. Generate statistics of the results.

Datasets

Forecast: Dummy text files found in the MET shared directory
Observation: Dummy text files found in the MET shared directory

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here for the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 434) section for more information.

METplus Components

This use case utilizes the METplus GridStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool grid_stat if all required files are found.

METplus Workflow

GridStat is the only tool called in this example. It processes a single run time with three forecast leads. The input data are simple text files with no timing information, so the list of forecast leads simply duplicates the same file multiple times to demonstrate how data is read in via Python embedding.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/GridStat/GridStat_python_embedding.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GridStat/GridStat_
↪python_embedding.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
```

(continues on next page)

(continued from previous page)

```
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 12H

LEAD_SEQ = 12


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###


FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/new
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY


GRID_STAT_CLIMO_MEAN_INPUT_DIR =
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/GridStat_python_embedding
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}

FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0
OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0


###
```

(continues on next page)

(continued from previous page)

```
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = FCST
OBTYPE = OBS

FCST_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py {INPUT_BASE}
↪/met_test/data/python/fcst.txt FCST
FCST_IS_PROB = false
FCST_GRID_STAT_PROB_THRESH = ==0.1

OBS_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py {INPUT_BASE}/
↪met_test/data/python/obs.txt OBS

GRID_STAT_ONCE_PER_FIELD = False


###
# GridStat Settings (optional)
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

#LOG_GRID_STAT_VERBOSITY = 2

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

GRID_STAT_REGRID_TO_GRID = G130

GRID_STAT_DESC = NA

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_OUTPUT_PREFIX = {MODEL}_vs_{OBTYPE}

GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_ECLV = BOTH
GRID_STAT_OUTPUT_FLAG_GRAD = BOTH

GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE

GRID_STAT_VERIFICATION_MASK_TEMPLATE =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
```

(continues on next page)

(continued from previous page)

```
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case calls a Python script to read the input data. The Python script is stored in the MET repository: /path/to/MET/installation/share/met/python/read_ascii_numpy.py

read_ascii_numpy.py

Running METplus

This use case can be run two ways:

1) Passing in GridStat_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat_
↪python_embedding.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_python_embedding.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat_
↪python_embedding.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/GridStat_python_embedding/2005080700 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_FCST_vs_OBS_120000L_20050807_120000V_eclv.txt
- grid_stat_FCST_vs_OBS_120000L_20050807_120000V_grad.txt
- grid_stat_FCST_vs_OBS_120000L_20050807_120000V.stat

Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-GridStat.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.11.2 GridStat: Multiple Config Files Use Case

met_tool_wrapper/GridStat/GridStat_multiple_config _files.conf

Scientific Objective

Compare 3 hour forecast precipitation accumulations to observations of 3 hour precipitation accumulation. Generate statistics of the results. Separate configuration files containing information about the forecast and observation data are passed into the METplus wrappers to demonstrate how users can create configuration files specific to their data sets to mix and match.

Datasets

Forecast: WRF 3 hour precipitation accumulation
Observation: MU 3 hour precipitation accumulation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here for the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See the *Running METplus* (page 448) section for more information.

METplus Components

This use case utilizes the METplus GridStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool GridStat if all required files are found.

METplus Workflow

GridStat is the only tool called in this example. It processes the following run times:

Init: 2005-08-07_0Z
Forecast lead: 12 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/GridStat/GridStat.conf

GridStat.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GridStat/GridStat.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide
```

(continues on next page)

(continued from previous page)

```
###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 12H

LEAD_SEQ = 12


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_GRID_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH}.tm00_G212

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/new
OBS_GRID_STAT_INPUT_TEMPLATE = ST2ml{valid?fmt=%Y%m%d%H}_A03h.nc

GRID_STAT_CLIMO_MEAN_INPUT_DIR =
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/GridStat/GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = WRF
OBTYPE = MC_PCP

GRID_STAT_ONCE_PER_FIELD = False

FCST_IS_PROB = false
#FCST_GRID_STAT_PROB_THRESH = ==0.1

FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A03
FCST_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2

OBS_VAR1_NAME = APCP_03
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2


###
# GridStat Settings (optional)
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

#LOG_GRID_STAT_VERBOSITY = 2

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

#FCST_GRID_STAT_FILE_TYPE =
#OBS_GRID_STAT_FILE_TYPE =

GRID_STAT_REGRID_TO_GRID = NONE
#GRID_STAT_REGRID_METHOD =
#GRID_STAT_REGRID_WIDTH =
#GRID_STAT_REGRID_VLD_THRESH =
#GRID_STAT_REGRID_SHAPE =
```

(continues on next page)

(continued from previous page)

```
#GRID_STAT_REGRID_CONVERT =
#GRID_STAT_REGRID_CENSOR_THRESH =
#GRID_STAT_REGRID_CENSOR_VAL =

#GRID_STAT_INTERP_FIELD =
#GRID_STAT_INTERP_VLD_THRESH =
#GRID_STAT_INTERP_SHAPE =
#GRID_STAT_INTERP_TYPE_METHOD =
#GRID_STAT_INTERP_TYPE_WIDTH =

#GRID_STAT_NC_PAIRS_VAR_NAME =

#GRID_STAT_CLIMO_MEAN_TIME_INTERP_METHOD =
#GRID_STAT_CLIMO_STDEV_TIME_INTERP_METHOD =

#GRID_STAT_GRID_WEIGHT_FLAG =

GRID_STAT_DESC = NA

FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0
OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}

#GRID_STAT_CLIMO_MEAN_FILE_NAME =
#GRID_STAT_CLIMO_MEAN_FIELD =
#GRID_STAT_CLIMO_MEAN_REGRID_METHOD =
#GRID_STAT_CLIMO_MEAN_REGRID_WIDTH =
#GRID_STAT_CLIMO_MEAN_REGRID_VLD_THRESH =
#GRID_STAT_CLIMO_MEAN_REGRID_SHAPE =
#GRID_STAT_CLIMO_MEAN_TIME_INTERP_METHOD =
#GRID_STAT_CLIMO_MEAN_MATCH_MONTH =
#GRID_STAT_CLIMO_MEAN_DAY_INTERVAL =
#GRID_STAT_CLIMO_MEAN_HOUR_INTERVAL =

#GRID_STAT_CLIMO_STDEV_FILE_NAME =
#GRID_STAT_CLIMO_STDEV_FIELD =
#GRID_STAT_CLIMO_STDEV_REGRID_METHOD =
```

(continues on next page)

(continued from previous page)

```
#GRID_STAT_CLIMO_STDEV_REGRID_WIDTH =
#GRID_STAT_CLIMO_STDEV_REGRID_VLD_THRESH =
#GRID_STAT_CLIMO_STDEV_REGRID_SHAPE =
#GRID_STAT_CLIMO_STDEV_TIME_INTERP_METHOD =
#GRID_STAT_CLIMO_STDEV_MATCH_MONTH =
#GRID_STAT_CLIMO_STDEV_DAY_INTERVAL =
#GRID_STAT_CLIMO_STDEV_HOUR_INTERVAL =


#GRID_STAT_CLIMO_CDF_BINS = 1
#GRID_STAT_CLIMO_CDF_CENTER_BINS = False
#GRID_STAT_CLIMO_CDF_WRITE_BINS = True
#GRID_STAT_CLIMO_CDF_DIRECT_PROB =

#GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
#GRID_STAT_OUTPUT_FLAG_MCTC = NONE
#GRID_STAT_OUTPUT_FLAG_MCTS = NONE
#GRID_STAT_OUTPUT_FLAG_CNT = NONE
#GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_SAL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_VL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_VAL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_VCNT = NONE
#GRID_STAT_OUTPUT_FLAG_PCT = NONE
#GRID_STAT_OUTPUT_FLAG_PSTD = NONE
#GRID_STAT_OUTPUT_FLAG_PJC = NONE
#GRID_STAT_OUTPUT_FLAG_PRC = NONE
GRID_STAT_OUTPUT_FLAG_ECLV = BOTH
#GRID_STAT_OUTPUT_FLAG_NBRCTC = NONE
#GRID_STAT_OUTPUT_FLAG_NBRCTS = NONE
#GRID_STAT_OUTPUT_FLAG_NBRCNT = NONE
GRID_STAT_OUTPUT_FLAG_GRAD = BOTH
#GRID_STAT_OUTPUT_FLAG_DMAP = NONE
#GRID_STAT_OUTPUT_FLAG_SEEPS =


GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
#GRID_STAT_NC_PAIRS_FLAG_CLIMO_CDP = FALSE
#GRID_STAT_NC_PAIRS_FLAG_WEIGHT = FALSE
#GRID_STAT_NC_PAIRS_FLAG_NBRHD = FALSE
```

(continues on next page)

(continued from previous page)

```
#GRID_STAT_NC_PAIRS_FLAG_FOURIER = FALSE
#GRID_STAT_NC_PAIRS_FLAG_GRADIENT = FALSE
#GRID_STAT_NC_PAIRS_FLAG_DISTANCE_MAP = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE
#GRID_STAT_NC_PAIRS_FLAG_SEEPS =

#GRID_STAT_SEEPS_P1_THRESH =

#GRID_STAT_HSS_EC_VALUE =

#GRID_STAT_MASK_GRID =
#GRID_STAT_MASK_POLY =

#GRID_STAT_DISTANCE_MAP_BADDELEY_P =
#GRID_STAT_DISTANCE_MAP_BADDELEY_MAX_DIST =
#GRID_STAT_DISTANCE_MAP_FOM_ALPHA =
#GRID_STAT_DISTANCE_MAP_ZHU_WEIGHT =
#GRID_STAT_DISTANCE_MAP_BETA_VALUE_N =

#GRID_STAT_FOURIER_WAVE_1D_BEG =
#GRID_STAT_FOURIER_WAVE_1D_END =

#GRID_STAT_CENSOR_THRESH =
#GRID_STAT_CENSOR_VAL =
```

GridStat_forecast.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GridStat/GridStat_
↪multiple_config_files.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_GRID_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH}.tm00_G212
```

(continues on next page)

(continued from previous page)

```
###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = WRF

FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A03
FCST_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2

FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0

FCST_IS_PROB = false
```

GridStat_observation.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GridStat/GridStat_
↪multiple_config_files.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/new
OBS_GRID_STAT_INPUT_TEMPLATE = ST2ml{valid?fmt=%Y%m%d%H}_A03h.nc

OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

OBTYPE = MC_PCP
```

(continues on next page)

(continued from previous page)

```
OBS_VAR1_NAME = APCP_03
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2

OBS_IS_PROB = false
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
```

(continues on next page)

(continued from previous page)

```
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
```

(continues on next page)

(continued from previous page)

```
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
```

(continues on next page)

(continued from previous page)

```
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat.conf, GridStat_forecast.conf, GridStat_observation.conf, an explicit override of the output directory, then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat.
↪conf
-c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat_forecast.conf
-c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat_observation.conf
-c dir.GRID_STAT_OUTPUT_DIR={OUTPUT_BASE}/met_tool_wrapper/GridStat/GridStat_multiple_
↪config
-c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, passing in GridStat.conf, GridStat_forecast.conf, GridStat_observation.conf, and an explicit override of the output directory:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat.
↪conf
-c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat_forecast.conf
-c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat_observation.conf
-c dir.GRID_STAT_OUTPUT_DIR={OUTPUT_BASE}/met_tool_wrapper/GridStat/GridStat_multiple_
↪config
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Note: The order that the configurations files are supplied on the command line is very important. If the same variables are found in multiple configuration files, then each subsequent configuration file will override the values of the previous files.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/GridStat/GridStat_multiple_config//2005080700 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_WRF_APCP_vs_MC_PCP_APCP_03_120000L_20050807_120000V_eclv.txt
- grid_stat_WRF_APCP_vs_MC_PCP_APCP_03_120000L_20050807_120000V_grad.txt
- grid_stat_WRF_APCP_vs_MC_PCP_APCP_03_120000L_20050807_120000V.stat

Keywords

Note:

- GridStatToolUseCase
- MultiConfUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-GridStat.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.11.3 GridStat: Basic Use Case

met_tool_wrapper/GridStat/GridStat.conf

Scientific Objective

Compare 3 hour forecast precipitation accumulations to observations of 3 hour precipitation accumulation. Generate statistics of the results.

Datasets

Forecast: WRF 3 hour precipitation accumulation
Observation: MU 3 hour precipitation accumulation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here for the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See the *Running METplus* (page 460) section for more information.

METplus Components

This use case utilizes the METplus GridStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool grid_stat if all required files are found.

METplus Workflow

GridStat is the only tool called in this example. It processes the following run times:

Init: 2005-08-07_0Z
Forecast lead: 12 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/GridStat/GridStat.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/GridStat/GridStat.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
```

(continues on next page)

(continued from previous page)

```
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 12H

LEAD_SEQ = 12


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_GRID_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH}.tm00_G212

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/new
OBS_GRID_STAT_INPUT_TEMPLATE = ST2ml{valid?fmt=%Y%m%d%H}_A03h.nc

GRID_STAT_CLIMO_MEAN_INPUT_DIR =
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/GridStat/GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = WRF
OBTYPE = MC_PCP

GRID_STAT_ONCE_PER_FIELD = False

FCST_IS_PROB = false
#FCST_GRID_STAT_PROB_THRESH = ==0.1

FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A03
FCST_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2

OBS_VAR1_NAME = APCP_03
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2


###
# GridStat Settings (optional)
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

#LOG_GRID_STAT_VERBOSITY = 2

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

#FCST_GRID_STAT_FILE_TYPE =
#OBS_GRID_STAT_FILE_TYPE =

GRID_STAT_REGRID_TO_GRID = NONE
#GRID_STAT_REGRID_METHOD =
#GRID_STAT_REGRID_WIDTH =
#GRID_STAT_REGRID_VLD_THRESH =
#GRID_STAT_REGRID_SHAPE =
#GRID_STAT_REGRID_CONVERT =
```

(continues on next page)

(continued from previous page)

```
#GRID_STAT_REGRID_CENSOR_THRESH =
#GRID_STAT_REGRID_CENSOR_VAL =

#GRID_STAT_INTERP_FIELD =
#GRID_STAT_INTERP_VLD_THRESH =
#GRID_STAT_INTERP_SHAPE =
#GRID_STAT_INTERP_TYPE_METHOD =
#GRID_STAT_INTERP_TYPE_WIDTH =

#GRID_STAT_NC_PAIRS_VAR_NAME =

#GRID_STAT_CLIMO_MEAN_TIME_INTERP_METHOD =
#GRID_STAT_CLIMO_STDEV_TIME_INTERP_METHOD =

#GRID_STAT_GRID_WEIGHT_FLAG =

GRID_STAT_DESC = NA

FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0
OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}

#GRID_STAT_CLIMO_MEAN_FILE_NAME =
#GRID_STAT_CLIMO_MEAN_FIELD =
#GRID_STAT_CLIMO_MEAN_REGRID_METHOD =
#GRID_STAT_CLIMO_MEAN_REGRID_WIDTH =
#GRID_STAT_CLIMO_MEAN_REGRID_VLD_THRESH =
#GRID_STAT_CLIMO_MEAN_REGRID_SHAPE =
#GRID_STAT_CLIMO_MEAN_TIME_INTERP_METHOD =
#GRID_STAT_CLIMO_MEAN_MATCH_MONTH =
#GRID_STAT_CLIMO_MEAN_DAY_INTERVAL =
#GRID_STAT_CLIMO_MEAN_HOUR_INTERVAL =

#GRID_STAT_CLIMO_STDEV_FILE_NAME =
#GRID_STAT_CLIMO_STDEV_FIELD =
#GRID_STAT_CLIMO_STDEV_REGRID_METHOD =
#GRID_STAT_CLIMO_STDEV_REGRID_WIDTH =
```

(continues on next page)

(continued from previous page)

```
#GRID_STAT_CLIMO_STDEV_REGRID_VLD_THRESH =
#GRID_STAT_CLIMO_STDEV_REGRID_SHAPE =
#GRID_STAT_CLIMO_STDEV_TIME_INTERP_METHOD =
#GRID_STAT_CLIMO_STDEV_MATCH_MONTH =
#GRID_STAT_CLIMO_STDEV_DAY_INTERVAL =
#GRID_STAT_CLIMO_STDEV_HOUR_INTERVAL =


#GRID_STAT_CLIMO_CDF_BINS = 1
#GRID_STAT_CLIMO_CDF_CENTER_BINS = False
#GRID_STAT_CLIMO_CDF_WRITE_BINS = True
#GRID_STAT_CLIMO_CDF_DIRECT_PROB =

#GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
#GRID_STAT_OUTPUT_FLAG_MCTC = NONE
#GRID_STAT_OUTPUT_FLAG_MCTS = NONE
#GRID_STAT_OUTPUT_FLAG_CNT = NONE
#GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_SAL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_VL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_VAL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_VCNT = NONE
#GRID_STAT_OUTPUT_FLAG_PCT = NONE
#GRID_STAT_OUTPUT_FLAG_PSTD = NONE
#GRID_STAT_OUTPUT_FLAG_PJC = NONE
#GRID_STAT_OUTPUT_FLAG_PRC = NONE
GRID_STAT_OUTPUT_FLAG_ECLV = BOTH
#GRID_STAT_OUTPUT_FLAG_NBRCTC = NONE
#GRID_STAT_OUTPUT_FLAG_NBRCTS = NONE
#GRID_STAT_OUTPUT_FLAG_NBRCNT = NONE
GRID_STAT_OUTPUT_FLAG_GRAD = BOTH
#GRID_STAT_OUTPUT_FLAG_DMAP = NONE
#GRID_STAT_OUTPUT_FLAG_SEEPS =


GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
#GRID_STAT_NC_PAIRS_FLAG_CLIMO_CDP = FALSE
#GRID_STAT_NC_PAIRS_FLAG_WEIGHT = FALSE
#GRID_STAT_NC_PAIRS_FLAG_NBRHD = FALSE
#GRID_STAT_NC_PAIRS_FLAG_FOURIER = FALSE
```

(continues on next page)

(continued from previous page)

```
#GRID_STAT_NC_PAIRS_FLAG_GRADIENT = FALSE
#GRID_STAT_NC_PAIRS_FLAG_DISTANCE_MAP = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE
#GRID_STAT_NC_PAIRS_FLAG_SEEPS =

#GRID_STAT_SEEPS_P1_THRESH =

#GRID_STAT_HSS_EC_VALUE =

#GRID_STAT_MASK_GRID =
#GRID_STAT_MASK_POLY =

#GRID_STAT_DISTANCE_MAP_BADDELEY_P =
#GRID_STAT_DISTANCE_MAP_BADDELEY_MAX_DIST =
#GRID_STAT_DISTANCE_MAP_FOM_ALPHA =
#GRID_STAT_DISTANCE_MAP_ZHU_WEIGHT =
#GRID_STAT_DISTANCE_MAP_BETA_VALUE_N =

#GRID_STAT_FOURIER_WAVE_1D_BEG =
#GRID_STAT_FOURIER_WAVE_1D_END =

#GRID_STAT_CENSOR_THRESH =
#GRID_STAT_CENSOR_VAL =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}
```

(continues on next page)

(continued from previous page)

```
tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat.
↪conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/GridStat.
↪conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/GridStat/GridStat/2005080700 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_WRF_APCP_vs_MC_PCP_APCP_03_120000L_20050807_120000V_eclv.txt
- grid_stat_WRF_APCP_vs_MC_PCP_APCP_03_120000L_20050807_120000V_grad.txt
- grid_stat_WRF_APCP_vs_MC_PCP_APCP_03_120000L_20050807_120000V.stat

Keywords

Note:

- GridStatToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-GridStat.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.12 IODA2NC

7.1.32.12.1 IODA2NC: Basic Use Case

met_tool_wrapper/IODA2NC/IODA2NC.conf

Scientific Objective

Convert IODA NetCDF files to MET NetCDF format.

Datasets

Input: IODA NetCDF observation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See the *Running METplus* (page 467) section for more information.

METplus Components

This use case utilizes the METplus IODA2NC wrapper to generate a command to run the MET tool ioda2nc if all required files are found.

METplus Workflow

IODA2NC is the only tool called in this example. It processes the following run time(s):

Valid: 2020-03-10 12Z

METplus Configuration

parm/use_cases/met_tool_wrapper/IODA2NC/IODA2NC.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/IODA2NC/IODA2NC.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = IODA2NC


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
```

(continues on next page)

(continued from previous page)

```
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2020031012
VALID_END = 2020031012
VALID_INCREMENT = 6H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

IODA2NC_INPUT_DIR = {INPUT_BASE}/met_test/new/ioda
IODA2NC_INPUT_TEMPLATE = ioda.NC001007.{valid?fmt=%Y%m%d%H}.nc

IODA2NC_OUTPUT_DIR = {OUTPUT_BASE}/ioda2nc
IODA2NC_OUTPUT_TEMPLATE = ioda.NC001007.{valid?fmt=%Y%m%d%H}.summary.nc



###
# IODA2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ioda2nc
###

#IODA2NC_VALID_BEG = {valid?fmt=%Y%m%d_%H?shift=-24H}
#IODA2NC_VALID_END = {valid?fmt=%Y%m%d_%H}
#IODA2NC_NMSG = 10

#IODA2NC_MESSAGE_TYPE =

#IODA2NC_MESSAGE_TYPE_MAP =

#IODA2NC_MESSAGE_TYPE_GROUP_MAP =

#IODA2NC_STATION_ID =
```

(continues on next page)

(continued from previous page)

```
IODA2NC_OBS_WINDOW_BEG = -5400
IODA2NC_OBS_WINDOW_END = 5400

#IODA2NC_MASK_GRID =
#IODA2NC_MASK_POLY =

IODA2NC_ELEVATION_RANGE_BEG = -1000
IODA2NC_ELEVATION_RANGE_END = 100000

#IODA2NC_LEVEL_RANGE_BEG = 1
#IODA2NC_LEVEL_RANGE_END = 255

#IODA2NC_OBS_VAR =

IODA2NC_OBS_NAME_MAP =
   { key = "wind_direction"; val = "WDIR"; },
   { key = "wind_speed";     val = "WIND"; }

#IODA2NC_METADATA_MAP =

#IODA2NC_MISSING_THRESH = <=-1e9, >=1e9, ==-9999

IODA2NC_QUALITY_MARK_THRESH = 0

IODA2NC_TIME_SUMMARY_FLAG = True
IODA2NC_TIME_SUMMARY_RAW_DATA = True
IODA2NC_TIME_SUMMARY_BEG = 000000
IODA2NC_TIME_SUMMARY_END = 235959
IODA2NC_TIME_SUMMARY_STEP = 300
IODA2NC_TIME_SUMMARY_WIDTH = 600
IODA2NC_TIME_SUMMARY_GRIB_CODE =
IODA2NC_TIME_SUMMARY_OBS_VAR = "WIND"
IODA2NC_TIME_SUMMARY_TYPE = "min", "max", "range", "mean", "stdev", "median", "p80"
IODA2NC_TIME_SUMMARY_VLD_FREQ = 0
IODA2NC_TIME_SUMMARY_VLD_THRESH = 0.0
```

MET Configuration

Note: See the *IODA2NC MET Configuration* (page 163) section of the User's Guide for more information on the environment variables used in the file below.

parm/met_config/IODA2NCConfig_wrapped

```
////////////////////////////////////////////////////////////////////////////////
//
// IODA2NC configuration file.
//
// For additional information, please see the MET Users Guide.
//
////////////////////////////////////////////////////////////////////////////////

//
// IODA message type
//
// message_type = [
${METPLUS_MESSAGE_TYPE}

//
// Mapping of message type group name to comma-separated list of values
// Derive PRMSL only for SURFACE message types
//
// message_type_group_map = [
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

//
// Mapping of input IODA message types to output message types
//
// message_type_map = [
${METPLUS_MESSAGE_TYPE_MAP}

//
// IODA station ID
//
// station_id = [
${METPLUS_STATION_ID}

////////////////////////////////////////////////////////////////////////////////

//
// Observation time window
//
```

(continues on next page)

(continued from previous page)

```
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observation retention regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observing location elevation
//
// elevation_range = {
${METPLUS_ELEVATION_RANGE_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Vertical levels to retain
//
// level_range = {
${METPLUS_LEVEL_RANGE_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// IODA variable names to retain or derive.
// Use obs_bufr_map to rename variables in the output.
// If empty or 'all', process all available variables.
//
// obs_var = [
${METPLUS_OBS_VAR}

////////////////////////////////////////////////////////////////////////////////

//
// Mapping of input IODA variable names to output variables names.
// The default IODA map, obs_var_map, is appended to this map.
//
// obs_name_map = [
${METPLUS_OBS_NAME_MAP}
```

(continues on next page)

(continued from previous page)

```
//
// Default mapping for Metadata.
//
// metadata_map = [
${METPLUS_METADATA_MAP}

// missing_thresh = [
${METPLUS_MISSING_THRESH}

////////////////////////////////////////////////////////////////////////////////

// quality_mark_thresh =
${METPLUS_QUALITY_MARK_THRESH}

////////////////////////////////////////////////////////////////////////////////

//
// Time periods for the summarization
// obs_var (string array) is added and works like grib_code (int array)
// when use_var_id is enabled and variable names are saved.
//
// time_summary = {
${METPLUS_TIME_SUMMARY_DICT}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

//version = "V10.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

Provide the use case .conf configuration file to the run_metplus.py script.

/path/to/METplus/parm/use_cases/met_tool_wrapper/IODA2NC/IODA2NC.conf

See the *Running METplus* (page 26) section of the System Configuration chapter for more details.

Expected Output

A successful run will output the following to the screen and the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/ioda2nc (relative to **OUTPUT_BASE**) and will contain the following file(s):

- ioda.NC001007.2020031012.summary.nc

Keywords

Note:

- IODA2NCToolUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1997) to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-IODA2NC.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.13 METdbLoad

7.1.32.13.1 METdbLoad: Basic Use Case

met_tool_wrapper/METdbLoad/METdbLoad.conf

Scientific Objective

Load MET data into a database using the met_db_load.py script found in dtcenter/METdataio

Datasets

Input: Various MET .stat and .tcst files

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to see the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 472) section for more information.

METplus Components

This use case utilizes the METplus METdbLoad wrapper to search for files ending with .stat or .tcst, substitute values into an XML load configuration file, and call met_db_load.py to load MET data into a database.

METplus Workflow

METdbLoad is the only tool called in this example. It does not loop over multiple run times:

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/METdbLoad/METdbLoad.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/METdbLoad/METdbLoad.
↪html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = METDbLoad


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
```

(continues on next page)

(continued from previous page)

```
###

MET_DB_LOAD_RUNTIME_FREQ = RUN_ONCE

LOOP_BY = VALID

VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2005080712
VALID_END = 2005080800
VALID_INCREMENT = 12H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

MET_DB_LOAD_INPUT_TEMPLATE = {INPUT_BASE}/met_test/out/grid_stat


###
# METdbLoad Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#metdbload
###

MET_DATA_DB_DIR = {METPLUS_BASE}/../METdataio

MET_DB_LOAD_XML_FILE = {PARM_BASE}/use_cases/met_tool_wrapper/METdbLoad/METdbLoadConfig.xml

MET_DB_LOAD_REMOVE_TMP_XML = True

MET_DB_LOAD_MV_HOST = localhost:3306
MET_DB_LOAD_MV_DATABASE = mv_metplus_test
MET_DB_LOAD_MV_USER = root
MET_DB_LOAD_MV_PASSWORD = mvuser

MET_DB_LOAD_MV_VERBOSE = false
MET_DB_LOAD_MV_INSERT_SIZE = 1
MET_DB_LOAD_MV_MODE_HEADER_DB_CHECK = false
MET_DB_LOAD_MV_DROP_INDEXES = false
MET_DB_LOAD_MV_APPLY_INDEXES = true
MET_DB_LOAD_MV_GROUP = METplus Input Test
MET_DB_LOAD_MV_LOAD_STAT = true
MET_DB_LOAD_MV_LOAD_MODE = false
```

(continues on next page)

(continued from previous page)

```
MET_DB_LOAD_MV_LOAD_MTD = false
MET_DB_LOAD_MV_LOAD_MPR = false
```

XML Configuration

METplus substitutes values in the template XML configuration file based on user settings in the METplus configuration file. While the XML template may appear to reference environment variables, this is not actually the case. These strings are used as a reference for the wrapper to substitute values.

Note: See the *METdbLoad XML Configuration* (page 169) section of the User's Guide for more information on the values substituted in the file below:

```
<load_spec>
  <connection>
    <host>${METPLUS_MV_HOST}</host>
    <database>${METPLUS_MV_DATABASE}</database>
    <user>${METPLUS_MV_USER}</user>
    <password>${METPLUS_MV_PASSWORD}</password>
  </connection>

  <verbose>${METPLUS_MV_VERBOSE}</verbose>
  <insert_size>${METPLUS_MV_INSERT_SIZE}</insert_size>
  <mode_header_db_check>${METPLUS_MV_MODE_HEADER_DB_CHECK}</mode_header_db_check>
  <drop_indexes>${METPLUS_MV_DROP_INDEXES}</drop_indexes>
  <apply_indexes>${METPLUS_MV_APPLY_INDEXES}</apply_indexes>
  <group>${METPLUS_MV_GROUP}</group>
  <load_stat>${METPLUS_MV_LOAD_STAT}</load_stat>
  <load_mode>${METPLUS_MV_LOAD_MODE}</load_mode>
  <load_mtd>${METPLUS_MV_LOAD_MTD}</load_mtd>
  <load_mpr>${METPLUS_MV_LOAD_MPR}</load_mpr>

  <folder_tmpl>{dirs}</folder_tmpl>
  <load_val>
    <field name="dirs">
      ${METPLUS_INPUT_PATHS}
    </field>
  </load_val>
</load_spec>
```

Running METplus

This use case can be run two ways:

1) Passing in METdbLoad.conf followed by a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/METdbLoad/METdbLoad.
↪conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config and then passing in METdbLoad.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/METdbLoad/METdbLoad.
↪conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path to directory where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Keywords

Note:

- METdbLoadUseCase
- AirQualityAndCompAppUseCase
- ClimateAppUseCase
- ShortRangeAppUseCase
- DataAssimilationAppUseCase
- EnsembleAppUseCase
- MarineAndCryosphereAppUseCase
- MediumRangeAppUseCase
- PrecipitationAppUseCase
- SpaceWeatherAppUseCase
- S2SAppUseCase
- S2SMJOAppUseCase
- S2SMidLatAppUseCase
- TCandExtraTCAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-METdbLoad.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.14 MODE

7.1.32.14.1 MODE: Using Python Embedding

met_tool_wrapper/MODE/MODE_python_embedding.conf

Scientific Objective

Compare dummy forecast data to dummy observations. Generate statistics of the results.

Datasets

Forecast: Dummy text files found in the MET shared directory
Observation: Dummy text files found in the MET shared directory

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

METplus Components

This use case utilizes the METplus MODE wrapper to search for files that are valid at a given run time and generate a command to run the MET tool mode if all required files are found.

METplus Workflow

MODE is the only tool called in this example. It processes a single run time with three forecast leads. The input data are simple text files with no timing information, so the list of forecast leads simply duplicates the same file multiple times to demonstrate how data is read in via Python embedding.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/MODE/MODE_python_embedding.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/MODE/MODE_python_
↪embedding.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide
```

(continues on next page)

(continued from previous page)

```
###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = MODE


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 12H

LEAD_SEQ = 12


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_MODE_INPUT_DIR =
FCST_MODE_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_MODE_INPUT_DIR =
OBS_MODE_INPUT_TEMPLATE = PYTHON_NUMPY

MODE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/MODE_python_embedding
MODE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}
```

(continues on next page)

(continued from previous page)

```
FCST_MODE_FILE_WINDOW_BEGIN = 0
FCST_MODE_FILE_WINDOW_END = 0
OBS_MODE_FILE_WINDOW_BEGIN = 0
OBS_MODE_FILE_WINDOW_END = 0


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

FCST_IS_PROB = false

FCST_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py {INPUT_BASE}
↪/met_test/data/python/fcst.txt FCST

FCST_MODE_CONV_RADIUS = 5
FCST_MODE_CONV_THRESH = >=80.0
FCST_MODE_MERGE_THRESH = >=75.0
FCST_MODE_MERGE_FLAG = NONE

OBS_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py {INPUT_BASE}/
↪met_test/data/python/obs.txt OBS
OBS_VAR1_LEVELS = P500

OBS_MODE_CONV_THRESH = >=80.0
OBS_MODE_CONV_RADIUS = 5
OBS_MODE_MERGE_THRESH = >=75.0
OBS_MODE_MERGE_FLAG = NONE


###
# MODE Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mode
###

#LOG_MODE_VERBOSITY = 2

MODE_CONFIG_FILE = {PARM_BASE}/met_config/MODEConfig_wrapped

MODE_REGRID_TO_GRID = NONE

MODE_MASK_MISSING_FLAG = BOTH

MODE_OUTPUT_PREFIX = FCST_vs_OBS
```

(continues on next page)

(continued from previous page)

```
MODE_MERGE_CONFIG_FILE =

MODEL = WRF

OBTYPE = WRF

#MODE_GRID_RES = 4

MODE_QUILT = True

MODE_VERIFICATION_MASK_TEMPLATE =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *MODE MET Configuration* (page 174) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
//
```

(continues on next page)

(continued from previous page)

```
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Approximate grid resolution (km)
//
// grid_res =
${METPLUS_GRID_RES}

////////////////////////////////////////////////////////////////////////////////

//
// Run all permutations of radius and threshold
//
// quilt =
${METPLUS_QUILT}

//
// MODE Multivar boolean combination logic
//
//multivar_logic =
${METPLUS_MULTIVAR_LOGIC}

//multivar_intensity_flag =
${METPLUS_MULTIVAR_INTENSITY_FLAG}

//
// Forecast and observation fields to be verified
//
```

(continues on next page)

(continued from previous page)

```
fcst = {
   ${METPLUS_FCST_FIELD}

   ${METPLUS_FCST_CENSOR_THRESH}
   ${METPLUS_FCST_CENSOR_VAL}
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}
   ${METPLUS_FCST_VLD_THRESH}
   ${METPLUS_FCST_FILTER_ATTR_NAME}
   ${METPLUS_FCST_FILTER_ATTR_THRESH}
   ${METPLUS_FCST_MERGE_THRESH}
   ${METPLUS_FCST_MERGE_FLAG}
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_MULTIVAR_NAME}
   ${METPLUS_FCST_MULTIVAR_LEVEL}
}

obs = {
   ${METPLUS_OBS_FIELD}

   ${METPLUS_OBS_CENSOR_THRESH}
   ${METPLUS_OBS_CENSOR_VAL}
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
   ${METPLUS_OBS_VLD_THRESH}
   ${METPLUS_OBS_FILTER_ATTR_NAME}
   ${METPLUS_OBS_FILTER_ATTR_THRESH}
   ${METPLUS_OBS_MERGE_THRESH}
   ${METPLUS_OBS_MERGE_FLAG}
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_MULTIVAR_NAME}
   ${METPLUS_OBS_MULTIVAR_LEVEL}
}

////////////////////////////////////////////////////////////////////////////////

//
// Handle missing data
//
// mask_missing_flag =
${METPLUS_MASK_MISSING_FLAG}

//
// Match objects between the forecast and observation fields
//
```

(continues on next page)

(continued from previous page)

```
//match_flag =
${METPLUS_MATCH_FLAG}

//
// Maximum centroid distance for objects to be compared
//
//max_centroid_dist =
${METPLUS_MAX_CENTROID_DIST}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine weights
//
//weight = {
${METPLUS_WEIGHT_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine interest functions
//
interest_function = {

   ${METPLUS_INTEREST_FUNCTION_CENTROID_DIST}

   ${METPLUS_INTEREST_FUNCTION_BOUNDARY_DIST}

   ${METPLUS_INTEREST_FUNCTION_CONVEX_HULL_DIST}

   angle_diff = (
      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 0.0 )
   );

   aspect_diff = (
```

(continues on next page)

(continued from previous page)

```
      (  0.00, 1.0 )
      (  0.10, 1.0 )
      (  0.75, 0.0 )
   );

   corner   = 0.8;
   ratio_if = (
      (    0.0, 0.0 )
      ( corner, 1.0 )
      (    1.0, 1.0 )
   );

   area_ratio = ratio_if;

   int_area_ratio = (
      ( 0.00, 0.00 )
      ( 0.10, 0.50 )
      ( 0.25, 1.00 )
      ( 1.00, 1.00 )
   );

   curvature_ratio = ratio_if;

   complexity_ratio = ratio_if;

   inten_perc_ratio = ratio_if;
}

////////////////////////////////////////////////////////////////////////////////

//
// Total interest threshold for determining matches
//
//total_interest_thresh =
${METPLUS_TOTAL_INTEREST_THRESH}

//
// Interest threshold for printing output pair information
//
print_interest_thresh = 0.0;

////////////////////////////////////////////////////////////////////////////////

//
// Plotting information
```

(continues on next page)

(continued from previous page)

```
//
met_data_dir = "MET_BASE";

fcst_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

obs_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

object_plot = {
   color_table      = "MET_BASE/colortables/mode_obj.ctable";
}

//
// Boolean for plotting on the region of valid data within the domain
//
plot_valid_flag = FALSE;

//
// Plot polyline edges using great circle arcs instead of straight lines
//
plot_gcarc_flag = FALSE;

////////////////////////////////////////////////////////////////////////////////

//
// NetCDF matched pairs, PostScript, and contingency table output files
//
//ps_plot_flag =
${METPLUS_PS_PLOT_FLAG}

//nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

//ct_stats_flag =
${METPLUS_CT_STATS_FLAG}


////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
shift_right = 0;   //  grid squares

////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case calls a Python script to read the input data. The Python script is stored in the MET repository: /path/to/MET/installation/share/met/python/read_ascii_numpy.py

read_ascii_numpy.py

Running METplus

This use case can be run two ways:

1) Passing in MODE_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/MODE/MODE_python_
↪embedding.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in MODE_python_embedding.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/MODE/MODE_python_
↪embedding.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/MODE_python_embedding/2005080712 (relative to **OUTPUT_BASE**) and will contain the following files: * mode_FCST_vs_OBS_120000L_20050807_120000V_120000A_cts.txt * mode_FCST_vs_OBS_120000L_20050807_120000V_120000A_obj.nc * mode_FCST_vs_OBS_120000L_20050807_12 * mode_FCST_vs_OBS_120000L_20050807_120000V_120000A.ps

Keywords

Note:

- MODEToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-MODE.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.14.2 MODE: Basic Use Case

met_tool_wrapper/MODE/MODE.conf

Scientific Objective

Compare relative humidity 12 hour forecast to 0 hour observations. Generate statistics of the results.

Datasets

Forecast: WRF Relative Humidity
Observation: WRF Relative Humidity

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 495) section for more information.

METplus Components

This use case utilizes the METplus MODE wrapper to search for files that are valid at a given run time and generate a command to run the MET tool mode if all required files are found.

METplus Workflow

MODE is the only tool called in this example. It processes the following run times:

Init: 2005-08-07_0Z
Forecast lead: 12 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/MODE/MODE.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/MODE/MODE.html
```

(continues on next page)

(continued from previous page)

```
# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = MODE


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 12H

LEAD_SEQ = 12


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_MODE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_MODE_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH}.tm00_G212

OBS_MODE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
OBS_MODE_INPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}/wrfprs_ruc13_00.tm00_G212
```

(continues on next page)

(continued from previous page)

```
MODE_OUTPUT_DIR = {OUTPUT_BASE}/mode
MODE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}

FCST_MODE_FILE_WINDOW_BEGIN = 0
FCST_MODE_FILE_WINDOW_END = 0

OBS_MODE_FILE_WINDOW_BEGIN = 0
OBS_MODE_FILE_WINDOW_END = 0

###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

FCST_IS_PROB = false

FCST_VAR1_NAME = RH
FCST_VAR1_LEVELS = P500

FCST_MODE_CONV_RADIUS = 5
FCST_MODE_CONV_THRESH = >=80.0
FCST_MODE_MERGE_THRESH = >=75.0
FCST_MODE_MERGE_FLAG = NONE
#MODE_FCST_FILE_TYPE =
#MODE_FCST_FILTER_ATTR_NAME =
#MODE_FCST_FILTER_ATTR_THRESH =
#MODE_FCST_CENSOR_THRESH =
#MODE_FCST_CENSOR_VAL =
#MODE_FCST_VLD_THRESH =

#MODE_FCST_MULTIVAR_NAME =
#MODE_FCST_MULTIVAR_LEVEL =


OBS_VAR1_NAME = RH
OBS_VAR1_LEVELS = P500

OBS_MODE_CONV_RADIUS = 5
OBS_MODE_CONV_THRESH = >=80.0
OBS_MODE_MERGE_THRESH = >=75.0
OBS_MODE_MERGE_FLAG = NONE

#MODE_OBS_FILE_TYPE =

#MODE_OBS_FILTER_ATTR_NAME =
```

(continues on next page)

(continued from previous page)

```
#MODE_OBS_FILTER_ATTR_THRESH =
#MODE_OBS_CENSOR_THRESH =
#MODE_OBS_CENSOR_VAL =
#MODE_OBS_VLD_THRESH =

#MODE_OBS_MULTIVAR_NAME =
#MODE_OBS_MULTIVAR_LEVEL =


###
# MODE Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mode
###

#LOG_MODE_VERBOSITY = 2

MODEL = WRF

MODE_DESC = NA

OBTYPE = WRF

MODE_CONFIG_FILE = {PARM_BASE}/met_config/MODEConfig_wrapped

#MODE_MULTIVAR_LOGIC =
#MODE_MULTIVAR_INTENSITY_FLAG =

MODE_REGRID_TO_GRID = NONE
#MODE_REGRID_METHOD =
#MODE_REGRID_WIDTH =
#MODE_REGRID_VLD_THRESH =
#MODE_REGRID_SHAPE =
#MODE_REGRID_CONVERT =
#MODE_REGRID_CENSOR_THRESH =
#MODE_REGRID_CENSOR_VAL =

MODE_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}_{CURRENT_OBS_
↪LEVEL}

MODE_MERGE_CONFIG_FILE =

MODE_GRID_RES = 40

#MODE_INTEREST_FUNCTION_CENTROID_DIST =
#MODE_INTEREST_FUNCTION_BOUNDARY_DIST =
```

(continues on next page)

(continued from previous page)

```
#MODE_INTEREST_FUNCTION_CONVEX_HULL_DIST =

#MODE_TOTAL_INTEREST_THRESH =

#MODE_MASK_GRID =
#MODE_MASK_GRID_FLAG =
#MODE_MASK_POLY =
#MODE_MASK_POLY_FLAG =

MODE_MASK_MISSING_FLAG = BOTH
#MODE_MATCH_FLAG =

#MODE_WEIGHT_CENTROID_DIST =
#MODE_WEIGHT_BOUNDARY_DIST =
#MODE_WEIGHT_CONVEX_HULL_DIST =
#MODE_WEIGHT_ANGLE_DIFF =
#MODE_WEIGHT_ASPECT_DIFF =
#MODE_WEIGHT_AREA_RATIO =
#MODE_WEIGHT_INT_AREA_RATIO =
#MODE_WEIGHT_CURVATURE_RATIO =
#MODE_WEIGHT_COMPLEXITY_RATIO =
#MODE_WEIGHT_INTEN_PERC_RATIO =
#MODE_WEIGHT_INTEN_PERC_VALUE =

#MODE_NC_PAIRS_FLAG_LATLON =
#MODE_NC_PAIRS_FLAG_RAW =
#MODE_NC_PAIRS_FLAG_OBJECT_RAW =
#MODE_NC_PAIRS_FLAG_OBJECT_ID =
#MODE_NC_PAIRS_FLAG_CLUSTER_ID =
#MODE_NC_PAIRS_FLAG_POLYLINES =

MODE_QUILT = True

#MODE_PS_PLOT_FLAG =
#MODE_CT_STATS_FLAG =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *MODE MET Configuration* (page 174) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Approximate grid resolution (km)
//
// grid_res =
${METPLUS_GRID_RES}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Run all permutations of radius and threshold
//
// quilt =
${METPLUS_QUILT}

//
// MODE Multivar boolean combination logic
//
//multivar_logic =
${METPLUS_MULTIVAR_LOGIC}

//multivar_intensity_flag =
${METPLUS_MULTIVAR_INTENSITY_FLAG}

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FIELD}

   ${METPLUS_FCST_CENSOR_THRESH}
   ${METPLUS_FCST_CENSOR_VAL}
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}
   ${METPLUS_FCST_VLD_THRESH}
   ${METPLUS_FCST_FILTER_ATTR_NAME}
   ${METPLUS_FCST_FILTER_ATTR_THRESH}
   ${METPLUS_FCST_MERGE_THRESH}
   ${METPLUS_FCST_MERGE_FLAG}
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_MULTIVAR_NAME}
   ${METPLUS_FCST_MULTIVAR_LEVEL}
}

obs = {
   ${METPLUS_OBS_FIELD}

   ${METPLUS_OBS_CENSOR_THRESH}
   ${METPLUS_OBS_CENSOR_VAL}
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
   ${METPLUS_OBS_VLD_THRESH}
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_OBS_FILTER_ATTR_NAME}
   ${METPLUS_OBS_FILTER_ATTR_THRESH}
   ${METPLUS_OBS_MERGE_THRESH}
   ${METPLUS_OBS_MERGE_FLAG}
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_MULTIVAR_NAME}
   ${METPLUS_OBS_MULTIVAR_LEVEL}
}

////////////////////////////////////////////////////////////////////////////////

//
// Handle missing data
//
// mask_missing_flag =
${METPLUS_MASK_MISSING_FLAG}

//
// Match objects between the forecast and observation fields
//
//match_flag =
${METPLUS_MATCH_FLAG}

//
// Maximum centroid distance for objects to be compared
//
//max_centroid_dist =
${METPLUS_MAX_CENTROID_DIST}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine weights
//
//weight = {
${METPLUS_WEIGHT_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine interest functions
//
interest_function = {

   ${METPLUS_INTEREST_FUNCTION_CENTROID_DIST}

   ${METPLUS_INTEREST_FUNCTION_BOUNDARY_DIST}

   ${METPLUS_INTEREST_FUNCTION_CONVEX_HULL_DIST}

   angle_diff = (
      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 0.0 )
   );

   aspect_diff = (
      (  0.00, 1.0 )
      (  0.10, 1.0 )
      (  0.75, 0.0 )
   );

   corner   = 0.8;
   ratio_if = (
      (    0.0, 0.0 )
      ( corner, 1.0 )
      (    1.0, 1.0 )
   );

   area_ratio = ratio_if;

   int_area_ratio = (
      ( 0.00, 0.00 )
      ( 0.10, 0.50 )
      ( 0.25, 1.00 )
      ( 1.00, 1.00 )
   );

   curvature_ratio = ratio_if;

   complexity_ratio = ratio_if;
```

(continues on next page)

(continued from previous page)

```
   inten_perc_ratio = ratio_if;
}

////////////////////////////////////////////////////////////////////////////////

//
// Total interest threshold for determining matches
//
//total_interest_thresh =
${METPLUS_TOTAL_INTEREST_THRESH}

//
// Interest threshold for printing output pair information
//
print_interest_thresh = 0.0;

////////////////////////////////////////////////////////////////////////////////

//
// Plotting information
//
met_data_dir = "MET_BASE";

fcst_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

obs_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

object_plot = {
   color_table      = "MET_BASE/colortables/mode_obj.ctable";
}

//
// Boolean for plotting on the region of valid data within the domain
//
plot_valid_flag = FALSE;

//
```

(continues on next page)

(continued from previous page)

```
// Plot polyline edges using great circle arcs instead of straight lines
//
plot_gcarc_flag = FALSE;

////////////////////////////////////////////////////////////////////////////////

//
// NetCDF matched pairs, PostScript, and contingency table output files
//
//ps_plot_flag =
${METPLUS_PS_PLOT_FLAG}

//nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

//ct_stats_flag =
${METPLUS_CT_STATS_FLAG}


////////////////////////////////////////////////////////////////////////////////

shift_right = 0;   //  grid squares

////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in MODE.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/MODE/MODE.conf -c /
↪path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in MODE.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/MODE/MODE.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in mode/2005080712 (relative to **OUTPUT_BASE**) and will contain the following files:

```
# * mode_WRF_RH_vs_WRF_RH_P500_120000L_20050807_120000V_000000A_cts.txt
# * mode_WRF_RH_vs_WRF_RH_P500_120000L_20050807_120000V_000000A_obj.nc
# * mode_WRF_RH_vs_WRF_RH_P500_120000L_20050807_120000V_000000A_obj.txt
# * mode_WRF_RH_vs_WRF_RH_P500_120000L_20050807_120000V_000000A.ps
```

Keywords

Note:

- MODEToolUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-MODE.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.15 MTD

7.1.32.15.1 MTD using Python Embedding

met_tool_wrapper/MTD/MTD_python_embedding.conf

Scientific Objective

Compare forecast and observation 3 hour precipitation accumulation spatially and temporally over the 6 hour, 9 hour, and 12 hour forecast leads.

Datasets

Forecast: Dummy text files found in the MET shared directory
Observation: Dummy text files found in the MET shared directory

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 506) section for more information.

METplus Components

This use case utilizes the METplus MTD wrapper to read in files using Python Embedding to demonstrate how to read in data this way.

METplus Workflow

MTD is the only tool called in this example. It processes a single run time with three forecast leads. The input data are simple text files with no timing information, so the list of forecast leads simply duplicates the same file multiple times to demonstrate how data is read in via Python Embedding.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/MTD/MTD_python_embedding.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/MTD/MTD_python_
↪embedding.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = MTD


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT=1M

LEAD_SEQ = 0, 1, 2


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

FCST_MTD_INPUT_DIR = {INPUT_BASE}/met_test/data/python
FCST_MTD_INPUT_TEMPLATE= fcst.txt

OBS_MTD_INPUT_DIR = {INPUT_BASE}/met_test/data/python
OBS_MTD_INPUT_TEMPLATE = obs.txt

MTD_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/MTD/mtd_python_embedding
MTD_OUTPUT_TEMPLATE =


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

FCST_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py MET_PYTHON_
↪INPUT_ARG FCST

FCST_MTD_CONV_RADIUS = 15
FCST_MTD_CONV_THRESH = >=5.0
FCST_MTD_MIN_VOLUME = 2000

FCST_MTD_INPUT_DATATYPE = PYTHON_NUMPY

FCST_IS_PROB = false
FCST_PROB_IN_GRIB_PDS = false

OBS_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py MET_PYTHON_
↪INPUT_ARG OBS

OBS_MTD_CONV_RADIUS = 15
OBS_MTD_CONV_THRESH = >=1.0

OBS_MTD_INPUT_DATATYPE = PYTHON_NUMPY


###
# MTD Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mtd
###

MTD_SINGLE_RUN = False
```

(continues on next page)

(continued from previous page)

```
MTD_SINGLE_DATA_SRC = OBS

MTD_CONFIG_FILE = {PARM_BASE}/met_config/MTDConfig_wrapped

MODEL = FCST
OBTYPE = OBS

#MTD_DESC =

MTD_REGRID_TO_GRID = OBS

MTD_OUTPUT_PREFIX = PYTHON
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *MTD MET Configuration* (page 188) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE Time Domain configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

   //
   // Output model name to be written
   //

${METPLUS_MODEL}

   //
   // Output description to be written
   //
```

(continues on next page)

(continued from previous page)

```
${METPLUS_DESC}

   //
   // Output observation type to be written
   //

${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

   //
   //  Verification grid
   //  May be set separately in each "field" entry
   //
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

   //
   // Approximate grid resolution (km)
   //

grid_res = 4;

////////////////////////////////////////////////////////////////////////////////

   //
   // Forecast and observation fields to be verified
   //

fcst = {

   ${METPLUS_FCST_FILE_TYPE}

   ${METPLUS_FCST_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}

}
```

(continues on next page)

(continued from previous page)

```
obs = {

   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
}

////////////////////////////////////////////////////////////////////////////////

   //
   // Intensity percentile value to be written
   //

inten_perc_value = 99;

////////////////////////////////////////////////////////////////////////////////

   //
   //  Throw away 3D objects with volumes smaller than this
   //
${METPLUS_MIN_VOLUME}


////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine weights
   //

weight = {

   space_centroid_dist  = 1.0;

   time_centroid_delta  = 1.0;

   speed_delta          = 1.0;

   direction_diff       = 1.0;
```

(continues on next page)

(continued from previous page)

```
   volume_ratio         = 1.0;

   axis_angle_diff      = 1.0;

   start_time_delta     = 1.0;

   end_time_delta       = 1.0;

}

////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine interest functions
   //

interest_function = {

   space_centroid_dist = (

      (   0.0, 1.0 )
      (  50.0, 0.5 )
      ( 100.0, 0.0 )

   );

   time_centroid_delta = (

      ( -3.0, 0.0 )
      ( -2.0, 0.5 )
      ( -1.0, 0.8 )
      (  0.0, 1.0 )
      (  1.0, 0.8 )
      (  2.0, 0.5 )
      (  3.0, 0.0 )

   );

   speed_delta = (

      ( -10.0, 0.0 )
      (  -5.0, 0.5 )
      (   0.0, 1.0 )
      (   5.0, 0.5 )
      (  10.0, 0.0 )
```

(continues on next page)

(continued from previous page)

```
);

direction_diff = (

   (   0.0, 1.0 )
   (  90.0, 0.0 )
   ( 180.0, 0.0 )

);

volume_ratio = (

   (  0.0, 0.0 )
   (  0.5, 0.5 )
   (  1.0, 1.0 )
   (  1.5, 0.5 )
   (  2.0, 0.0 )

);

axis_angle_diff = (

   (  0.0, 1.0 )
   ( 30.0, 1.0 )
   ( 90.0, 0.0 )

);

start_time_delta = (

   ( -5.0, 0.0 )
   ( -3.0, 0.5 )
   (  0.0, 1.0 )
   (  3.0, 0.5 )
   (  5.0, 0.0 )

);

end_time_delta = (

   ( -5.0, 0.0 )
   ( -3.0, 0.5 )
   (  0.0, 1.0 )
   (  3.0, 0.5 )
```

(continues on next page)

(continued from previous page)

```
      (  5.0, 0.0 )

   );

}   //  interest functions


////////////////////////////////////////////////////////////////////////////////

   //
   // Total interest threshold for determining matches
   //

total_interest_thresh = 0.7;


////////////////////////////////////////////////////////////////////////////////

   //
   // Output flags
   //

nc_output = {

   latlon       = true;
   raw          = true;
   object_id    = true;
   cluster_id   = true;

}

txt_output = {

   attributes_2d   = true;
   attributes_3d   = true;

}


////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V9.0";

tmp_dir = "${MET_TMP_DIR}";
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case calls a Python script to read the input data. The Python script is stored in the MET repository: /path/to/MET/installation/share/met/python/read_ascii_numpy.py

read_ascii_numpy.py

Running METplus

This use case can be run two ways:

1) Passing in MTD_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/MTD/MTD_python_
↪embedding.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in MTD_python_embedding.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/MTD/MTD_python_
↪embedding.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/MTD/mtd_python_embedding (relative to **OUTPUT_BASE**) and will contain the following files:

- mtd_PYTHON_20050807_120000V_2d.txt
- mtd_PYTHON_20050807_120000V_3d_pair_cluster.txt
- mtd_PYTHON_20050807_120000V_3d_pair_simple.txt
- mtd_PYTHON_20050807_120000V_3d_single_cluster.txt
- mtd_PYTHON_20050807_120000V_3d_single_simple.txt
- mtd_PYTHON_20050807_120000V_obj.nc

Keywords

Note:

- MTDToolUseCase
- PythonEmbeddingFileUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-MTD.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.15.2 Basic MTD Use Case

met_tool_wrappper/MTD/MTD.conf

Scientific Objective

Compare forecast and observation 3 hour precipitation accumulation spatially and temporally over the 6 hour, 9 hour, and 12 hour forecast leads.

Datasets

Forecast: WRF GRIB Precipitation Accumulation
Observation: Stage 2 NetCDF Precipitation Accumulation (converted from GRIB format)

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 517) section for more information.

METplus Components

This use case utilizes the METplus MTD wrapper to search for files that are valid at a given run time and generate a command to run the MET tool mode if all required files are found.

METplus Workflow

MTD is the only tool called in this example. It processes the following run times:

Init: 2005-08-07_0Z
Forecast leads: 6, 9, and 12 hours

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/MTD/MTD.conf

```
[config]

# Documentation for this use case can be found at
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/MTD/MTD.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = MTD


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT=1M

LEAD_SEQ = 6H, 9H, 12H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_MTD_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_MTD_INPUT_TEMPLATE= {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH}.tm00_G212

OBS_MTD_INPUT_DIR = {INPUT_BASE}/met_test/new
OBS_MTD_INPUT_TEMPLATE = ST2ml{valid?fmt=%Y%m%d%H}_A03h.nc
```

(continues on next page)

(continued from previous page)

```
MTD_OUTPUT_DIR = {OUTPUT_BASE}/mtd
MTD_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

FCST_IS_PROB = False
FCST_PROB_IN_GRIB_PDS = false

FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A03
FCST_VAR1_THRESH = gt12.7

FCST_MTD_CONV_RADIUS = 10
FCST_MTD_CONV_THRESH = >=0.0

OBS_VAR1_NAME = APCP_03
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_THRESH = gt12.7

OBS_MTD_CONV_RADIUS = 10
OBS_MTD_CONV_THRESH = >=0.0


###
# MTD Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mtd
###

MTD_SINGLE_RUN = False
MTD_SINGLE_DATA_SRC = OBS

MODEL = WRF
MTD_DESC = NA
OBTYPE = MC_PCP

MTD_CONFIG_FILE = {PARM_BASE}/met_config/MTDConfig_wrapped

MTD_REGRID_TO_GRID = OBS
#MTD_REGRID_METHOD =
#MTD_REGRID_WIDTH =
```

(continues on next page)

(continued from previous page)

```
#MTD_REGRID_VLD_THRESH =
#MTD_REGRID_SHAPE =
#MTD_REGRID_CONVERT =
#MTD_REGRID_CENSOR_THRESH =
#MTD_REGRID_CENSOR_VAL =


MTD_MIN_VOLUME = 2000

MTD_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}_{CURRENT_FCST_
↪LEVEL}
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *MTD MET Configuration* (page 188) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE Time Domain configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

   //
   // Output model name to be written
   //

${METPLUS_MODEL}

   //
   // Output description to be written
   //
```

(continues on next page)

(continued from previous page)

```
${METPLUS_DESC}

   //
   // Output observation type to be written
   //

${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

   //
   //  Verification grid
   //  May be set separately in each "field" entry
   //
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

   //
   // Approximate grid resolution (km)
   //

grid_res = 4;

////////////////////////////////////////////////////////////////////////////////

   //
   // Forecast and observation fields to be verified
   //

fcst = {

   ${METPLUS_FCST_FILE_TYPE}

   ${METPLUS_FCST_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}

}

obs = {
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
}

////////////////////////////////////////////////////////////////////////////////

   //
   // Intensity percentile value to be written
   //

inten_perc_value = 99;

////////////////////////////////////////////////////////////////////////////////

   //
   //  Throw away 3D objects with volumes smaller than this
   //
${METPLUS_MIN_VOLUME}


////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine weights
   //

weight = {

   space_centroid_dist  = 1.0;

   time_centroid_delta  = 1.0;

   speed_delta          = 1.0;

   direction_diff       = 1.0;

   volume_ratio         = 1.0;
```

(continues on next page)

(continued from previous page)

```
   axis_angle_diff      = 1.0;

   start_time_delta     = 1.0;

   end_time_delta       = 1.0;

}

////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine interest functions
   //

interest_function = {

   space_centroid_dist = (

      (   0.0, 1.0 )
      (  50.0, 0.5 )
      ( 100.0, 0.0 )

   );

   time_centroid_delta = (

      ( -3.0, 0.0 )
      ( -2.0, 0.5 )
      ( -1.0, 0.8 )
      (  0.0, 1.0 )
      (  1.0, 0.8 )
      (  2.0, 0.5 )
      (  3.0, 0.0 )

   );

   speed_delta = (

      ( -10.0, 0.0 )
      (  -5.0, 0.5 )
      (   0.0, 1.0 )
      (   5.0, 0.5 )
      (  10.0, 0.0 )
```

(continues on next page)

(continued from previous page)

```
);

direction_diff = (

   (   0.0, 1.0 )
   (  90.0, 0.0 )
   ( 180.0, 0.0 )

);

volume_ratio = (

   (  0.0, 0.0 )
   (  0.5, 0.5 )
   (  1.0, 1.0 )
   (  1.5, 0.5 )
   (  2.0, 0.0 )

);

axis_angle_diff = (

   (  0.0, 1.0 )
   ( 30.0, 1.0 )
   ( 90.0, 0.0 )

);

start_time_delta = (

   ( -5.0, 0.0 )
   ( -3.0, 0.5 )
   (  0.0, 1.0 )
   (  3.0, 0.5 )
   (  5.0, 0.0 )

);

end_time_delta = (

   ( -5.0, 0.0 )
   ( -3.0, 0.5 )
   (  0.0, 1.0 )
   (  3.0, 0.5 )
   (  5.0, 0.0 )
```

(continues on next page)

(continued from previous page)

```
   );

}   //  interest functions


////////////////////////////////////////////////////////////////////////////////

   //
   // Total interest threshold for determining matches
   //

total_interest_thresh = 0.7;


////////////////////////////////////////////////////////////////////////////////

   //
   // Output flags
   //

nc_output = {

   latlon       = true;
   raw          = true;
   object_id    = true;
   cluster_id   = true;

}

txt_output = {

   attributes_2d   = true;
   attributes_3d   = true;

}


////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V9.0";

tmp_dir = "${MET_TMP_DIR}";
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in MTD.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/MTD/MTD.conf -c /
↪path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in MTD.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/MTD/MTD.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in mtd/2005080712 (relative to **OUTPUT_BASE**) and will contain the following files:

```
# * mtd_PROB_WRF_APCP_vs_MC_PCP_APCP_03_A03_20050807_060000V_2d.txt
# * mtd_PROB_WRF_APCP_vs_MC_PCP_APCP_03_A03_20050807_060000V_3d_single_simple.txt
# * mtd_PROB_WRF_APCP_vs_MC_PCP_APCP_03_A03_20050807_060000V_obj.nc
```

Keywords

Note:

- MTDToolUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-MTD.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.16 PB2NC

7.1.32.16.1 PB2NC: Basic Use Case

met_tool_wrapper/PB2NC/PB2NC.conf

Scientific Objective

Simply converting file formats so point observations can be read by the MET tools.

Datasets

Observations: Various fields in prepBUFR file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 524) section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus PB2NC wrapper to generate a command to run the MET tool PB2NC if all required files are found.

METplus Workflow

PB2NC is the only tool called in this example. It processes the following run time:

Valid: 2007-03-31_12Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PB2NC/PB2NC.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PB2NC/PB2NC.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PB2NC


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
```

(continues on next page)

(continued from previous page)

```
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2007033112
VALID_END = 2007033112
VALID_INCREMENT = 1M

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

PB2NC_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_obs/prepbufr
PB2NC_INPUT_TEMPLATE = ndas.t{da_init?fmt=%H}z.prepbufr.tm{offset?fmt=%2H}.{da_init?fmt=%Y%m
↪%d}.nr

PB2NC_OFFSETS = 12

PB2NC_SKIP_IF_OUTPUT_EXISTS = True

PB2NC_OUTPUT_DIR = {OUTPUT_BASE}/pb2nc
PB2NC_OUTPUT_TEMPLATE = sample_pb.nc


###
# PB2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pb2nc
###

PB2NC_CONFIG_FILE = {PARM_BASE}/met_config/PB2NCConfig_wrapped

PB2NC_OBS_WINDOW_BEGIN = -1800
PB2NC_OBS_WINDOW_END = 1800

PB2NC_VALID_BEGIN = {valid?fmt=%Y%m%d_%H}
PB2NC_VALID_END = {valid?fmt=%Y%m%d_%H?shift=1d}

PB2NC_GRID = G212
PB2NC_POLY =
```

(continues on next page)

(continued from previous page)

```
PB2NC_STATION_ID =
PB2NC_MESSAGE_TYPE =

PB2NC_PB_REPORT_TYPE = 120, 220, 221, 122, 222, 223, 224, 131, 133, 233, 153, 156, 157, 180,␣
↪280, 181, 182, 281, 282, 183, 284, 187, 287

#PB2NC_LEVEL_RANGE_BEG =
#PB2NC_LEVEL_RANGE_END =

PB2NC_LEVEL_CATEGORY = 0, 1, 4, 5, 6

PB2NC_QUALITY_MARK_THRESH = 3

# Leave empty to process all
PB2NC_OBS_BUFR_VAR_LIST = QOB, TOB, ZOB, UOB, VOB, D_DPT, D_WIND, D_RH, D_MIXR

PB2NC_TIME_SUMMARY_FLAG = False
PB2NC_TIME_SUMMARY_BEG = 000000
PB2NC_TIME_SUMMARY_END = 235959
PB2NC_TIME_SUMMARY_VAR_NAMES =
PB2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80

PB2NC_TIME_SUMMARY_RAW_DATA = False
PB2NC_TIME_SUMMARY_STEP = 3600
PB2NC_TIME_SUMMARY_WIDTH = 3600
PB2NC_TIME_SUMMARY_GRIB_CODES =
PB2NC_TIME_SUMMARY_VALID_FREQ = 0
PB2NC_TIME_SUMMARY_VALID_THRESH = 0.0

#PB2NC_OBS_BUFR_MAP =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *PB2NC MET Configuration* (page 198) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// PB2NC configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// PrepBufr message type
//
${METPLUS_MESSAGE_TYPE}

//
// Mapping of message type group name to comma-separated list of values
// Derive PRMSL only for SURFACE message types
//
message_type_group_map = [
   { key = "SURFACE"; val = "ADPSFC,SFCSHP,MSONET";               },
   { key = "ANYAIR";  val = "AIRCAR,AIRCFT";                      },
   { key = "ANYSFC";  val = "ADPSFC,SFCSHP,ADPUPA,PROFLR,MSONET"; },
   { key = "ONLYSF";  val = "ADPSFC,SFCSHP";                      }
];

//
// Mapping of input PrepBufr message types to output message types
//
message_type_map = [];

//
// PrepBufr station ID
//
${METPLUS_STATION_ID}


////////////////////////////////////////////////////////////////////////////////

//
// Observation time window
//
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observation retention regions
```

(continues on next page)

(continued from previous page)

```
//
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observing location elevation
//
elevation_range = {
   beg =  -1000;
   end = 100000;
}

////////////////////////////////////////////////////////////////////////////////

//
// Observation types
//
//pb_report_type =
${METPLUS_PB_REPORT_TYPE}

in_report_type  = [];

instrument_type = [];

////////////////////////////////////////////////////////////////////////////////

//
// Vertical levels to retain
//
//level_range = {
${METPLUS_LEVEL_RANGE_DICT}

//level_category =
${METPLUS_LEVEL_CATEGORY}

////////////////////////////////////////////////////////////////////////////////

//
// BUFR variable names to retain or derive.
// If emtpy, process all available variables.
//
${METPLUS_OBS_BUFR_VAR}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Mapping of BUFR variable name to GRIB name. The default map is defined at
// obs_prepbufr_map. This replaces/expends the default map.
//
//obs_bufr_map =
${METPLUS_OBS_BUFR_MAP}

// This map is for PREPBUFR. It will be added into obs_bufr_map.
// Please do not override this map.
//obs_prepbufr_map =


////////////////////////////////////////////////////////////////////////////////

//quality_mark_thresh =
${METPLUS_QUALITY_MARK_THRESH}

event_stack_flag    = TOP;

////////////////////////////////////////////////////////////////////////////////
//
// Time periods for the summarization
//
${METPLUS_TIME_SUMMARY_DICT}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

//version = "V9.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in PB2NC.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PB2NC/PB2NC.conf -c /
↪path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PB2NC.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PB2NC/PB2NC.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in pb2nc (relative to **OUTPUT_BASE**) and will contain the following file:

- sample_pb.nc

Keywords

Note:

- PB2NCToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PB2NC.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.17 PCPCombine

7.1.32.17.1 PCPCombine: Python Embedding Use Case

met_tool_wrapper/PCPCombine/PCPCombine_python_embedding.conf

Scientific Objective

Build a 2 hour precipitation accumulation field from 30 minute IMERG data.

Datasets

Forecast: IMERG HDF5 30 minute precipitation accumulation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 529) section for more information.

Data Source: IMERG

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* h5-py
* numpy
```

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this it the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/h5-py/and/numpy/packages/bin/python

METplus Components

This use case utilizes the METplus PCPCombine wrapper to run a Python script to read input data to build the desired accumulation for a given run time using a filename template and a list of available input accumulations. If enough files meeting the criteria are found to build the output accumulation, it will generate a command to run PCPCombine to combine the data.

METplus Workflow

PCPCombine is the only tool called in this example. It processes the following run times:

Valid: 2018-01-02_13:30Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PCPCombine/PCPCombine_python_embedding.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PCPCombine/PCPCombine_
↪python_embedding.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PcpCombine


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
```

(continues on next page)

(continued from previous page)

```
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H%M
VALID_BEG=201801021300
VALID_END=201801021300
VALID_INCREMENT=43200

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

OBS_PCP_COMBINE_RUN = True

OBS_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/met_test/new/imerg
OBS_PCP_COMBINE_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/PCPCombine/PCPCombine_combine_py_
↪embed
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = IMERG.{valid?fmt=%Y%m%d_%H%M}_A{level?fmt=%2H}h


###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

#LOG_PCP_COMBINE_VERBOSITY = 2

OBS_PCP_COMBINE_METHOD = ADD

OBS_VAR1_NAME = APCP
OBS_VAR1_LEVELS = A06

OBS_PCP_COMBINE_INPUT_DATATYPE = PYTHON_NUMPY
OBS_PCP_COMBINE_INPUT_ACCUMS = 6
```

(continues on next page)

(continued from previous page)

```
OBS_PCP_COMBINE_INPUT_NAMES = {PARM_BASE}/use_cases/met_tool_wrapper/PCPCombine/sum_IMERG_
↪V06_HDF5.py {OBS_PCP_COMBINE_INPUT_DIR} IRprecipitation {valid?fmt=%Y%m%d%H} 02

[user_env_vars]
# uncomment and change this to the path of a version of python that has the h5py package␣
↪installed
#MET_PYTHON_EXE = /path/to/python/with/h5-py/and/numpy/packages/bin/python
```

MET Configuration

None. PCPCombine does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in PCPCombine_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_python_embedding.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PCPCombine_python_embedding.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_python_embedding.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/PCPCombine/PCPCombine_py_embed (relative to **OUTPUT_BASE**) and will contain the following files:

- .

Keywords

Note:

- PCPCombineToolUseCase
- PythonEmbeddingFileUseCase
- MET_PYTHON_EXEUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PCPCombine.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.17.2 PCPCombine: SUM Use Case

met_tool_wrapper/PCPCombine/PCPCombine_sum.conf

Scientific Objective

Build a 15 minute precipitation accumulation field from 5 minute accumulation fields.

Datasets

Forecast: NEWSe 5 minute precipitation accumulation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 533) section for more information.

Data Source: NEWSe

METplus Components

This use case utilizes the METplus PCPCombine wrapper to build a command that will look for valid data to build an accumulation.

METplus Workflow

PCPCombine is the only tool called in this example. It processes the following run times:

Valid: 2019-08-02_18:15Z
Forecast lead: 15 minute

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PCPCombine/PCPCombine_sum.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PCPCombine/PCPCombine_
↪sum.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PcpCombine
```

(continues on next page)

(continued from previous page)

```
###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H%M
VALID_BEG = 201908021815
VALID_END = 201908021815
VALID_INCREMENT = 1M

LEAD_SEQ = 15M


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_PCP_COMBINE_RUN = True

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/met_test/new
FCST_PCP_COMBINE_INPUT_TEMPLATE = NEWSe_{init?fmt=%Y%m%d}_i{init?fmt=%H%M}_m0_f*

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/PCPCombine/PCPCombine_sum
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = NEWSe5min_mem00_lag00.nc


###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

FCST_PCP_COMBINE_METHOD = SUM

#LOG_PCP_COMBINE_VERBOSITY = 2
```

(continues on next page)

(continued from previous page)

```
FCST_IS_PROB = false

FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB
FCST_PCP_COMBINE_INPUT_ACCUMS = 5M
FCST_PCP_COMBINE_INPUT_NAMES = A000500
FCST_PCP_COMBINE_INPUT_LEVELS = Surface

FCST_PCP_COMBINE_OUTPUT_ACCUM = 15M
FCST_PCP_COMBINE_OUTPUT_NAME = A001500
```

MET Configuration

None. PCPCombine does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in PCPCombine_sum.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_sum.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PCPCombine_sum.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_sum.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/PCPCombine/PCPCombine_sum (relative to **OUTPUT_BASE**) and will contain the following files:

- NEWSe5min_mem00_lag00.nc

Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PCPCombine.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.17.3 PCPCombine: DERIVE Use Case

met_tool_wrapper/PCPCombine/PCPCombine_derive.conf

Scientific Objective

Derive statistics (sum, minimum, maximum, range, mean, standard deviation, and valid count) using six 3 hour precipitation accumulation fields.

Datasets

Forecast: WRF precipitation accumulation fields (24, 21, 18, 15, 12, and 9 hour forecast leads)

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 537) section for more information.

Data Source: WRF

METplus Components

This use case utilizes the METplus PCPCombine wrapper to search for files for each run time using a filename template, forecast lead, and lookback time. It will a generate a command to run PCPCombine to derive statistics from the fields.

METplus Workflow

PCPCombine is the only tool called in this example. It processes the following run times:

Valid: 2005-08-07_00Z
Forecast lead: 24 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PCPCombine/PCPCombine_derive.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PCPCombine/PCPCombine_
↪derive.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PCPCombine
```

(continues on next page)

(continued from previous page)

```
###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2005080700
INIT_END = 2005080700
INIT_INCREMENT = 1M

LEAD_SEQ = 24H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_PCP_COMBINE_RUN = True

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_PCP_COMBINE_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH}.tm00_G212

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/PCPCombine/PCPCombine_derive
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = wrfprs_ruc13_{init?fmt=%Y%m%d%H}_f{lead?fmt=%HH}_A{level?
↪fmt=%HH}.nc


###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

#LOG_PCP_COMBINE_VERBOSITY = 2
```

(continues on next page)

(continued from previous page)

```
FCST_PCP_COMBINE_METHOD = DERIVE

FCST_PCP_COMBINE_STAT_LIST = sum,min,max,range,mean,stdev,vld_count

FCST_PCP_COMBINE_DERIVE_LOOKBACK = 18H

FCST_PCP_COMBINE_MIN_FORECAST = 9H
FCST_PCP_COMBINE_MAX_FORECAST = 2d

FCST_IS_PROB = false

FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB

FCST_PCP_COMBINE_INPUT_ACCUMS = 3H
FCST_PCP_COMBINE_INPUT_NAMES = APCP
FCST_PCP_COMBINE_INPUT_LEVELS = A03
FCST_PCP_COMBINE_INPUT_OPTIONS =

FCST_PCP_COMBINE_OUTPUT_ACCUM = 18H
FCST_PCP_COMBINE_OUTPUT_NAME =

#FCST_PCP_COMBINE_EXTRA_NAMES =
#FCST_PCP_COMBINE_EXTRA_LEVELS =
#FCST_PCP_COMBINE_EXTRA_OUTPUT_NAMES =
```

MET Configuration

None. PCPCombine does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in PCPCombine_derive.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_derive.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PCPCombine_derive.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_derive.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/PCPCombine/PCPCombine_derive (relative to **OUTPUT_BASE**) and will contain the following files:

- wrfprs_ruc13_2005080700_f24_A18.nc

Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PCPCombine.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.17.4 PCPCombine: SUBTRACT Use Case

met_tools_wrapper/PCPCombine/PCPCombine_subtract.conf

Scientific Objective

Extract a 3 hour precipitation accumulation field by subtracting a 15 hour accumulation field from an 18 hour accumulation field.

Datasets

Forecast: WRF precipitation accumulation fields (18 hour and 15 hour forecast leads)

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 542) section for more information.

Data Source: WRF

METplus Components

This use case utilizes the METplus PCPCombine wrapper to search for files to extract the desired accumulation for a given run time using a filename template, forecast lead, and output accumulation. It will a generate a command to run PCPCombine to subtract a field from another field to extract the desired accumulation.

METplus Workflow

PCPCombine is the only tool called in this example. It processes the following run times:

Valid: 2005-08-07_00Z
Forecast lead: 18 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PCPCombine/PCPCombine_subtract.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PCPCombine/PCPCombine_
↪subtract.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PcpCombine


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT

INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2005080700
INIT_END = 2005080700
INIT_INCREMENT = 1M

LEAD_SEQ = 18H


###
# File I/O
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_PCP_COMBINE_RUN = True

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_PCP_COMBINE_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH}.tm00_G212

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/PCPCombine/PCPCombine_subtract
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = wrfprs_ruc13_{init?fmt=%Y%m%d%H}_f{lead?fmt=%HH}_A03.nc


###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

#LOG_PCP_COMBINE_VERBOSITY = 2

FCST_PCP_COMBINE_METHOD = SUBTRACT

FCST_PCP_COMBINE_MAX_FORECAST = 2d

FCST_IS_PROB = false

FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB

FCST_PCP_COMBINE_OUTPUT_ACCUM = 3H

FCST_PCP_COMBINE_OUTPUT_NAME = APCP_03

FCST_PCP_COMBINE_USE_ZERO_ACCUM = False
```

MET Configuration

None. PCPCombine does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in PCPCombine_subtract.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_subtract.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PCPCombine_subtract.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_subtract.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/PCPCombine/PCPCombine_subtract (relative to **OUTPUT_BASE**) and will contain the following files:

- wrfprs_ruc13_2005080700_f24_A18.nc

Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PCPCombine.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.17.5 PCPCombine: User-defined Command Use Case

met_tool_wrapper/PCPCombine/PCPCombine_user_defined.conf

Scientific Objective

Derive statistics (sum, minimum, maximum, range, mean, standard deviation, and valid count) using six 3 hour precipitation accumulation fields. This use case builds the same command as pcp_derive.conf, but the command is defined completely by the user in the METplus configuration file.

Datasets

Forecast: WRF precipitation accumulation fields (24, 21, 18, 15, 12, and 9 hour forecast leads)

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 546) section for more information.

Data Source: WRF

METplus Components

This use case utilizes the METplus PCPCombine wrapper to generate a command to run PCPCombine to derive statistics from the fields. FCST_PCP_COMBINE_COMMAND is used to define all arguments to the call to the MET tool pcp_combine. This variable uses filename template notation using the 'shift' keyword to define filenames that are valid at a time slightly shifted from the run time, i.e. wrfprs_ruc13_{lead?fmt=%HH?shift=-3H}.tm00_G212. It also references other configuration variables in the METplus configuration file, such as FCST_PCP_COMBINE_INPUT_NAMES and FCST_PCP_COMBINE_INPUT_LEVELS, and FCST_PCP_COMBINE_INPUT_DIR.

METplus Workflow

PCPCombine is the only tool called in this example. It processes the following run times:

Valid: 2005-08-07_00Z
Forecast lead: 24 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PCPCombine/PCPCombine_user_defined.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PCPCombine/PCPCombine_
↪user_defined.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PcpCombine


###
# Time Info
```

(continues on next page)

(continued from previous page)

```
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###


LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2005080700
INIT_END = 2005080700
INIT_INCREMENT = 1M

LEAD_SEQ = 24H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_PCP_COMBINE_RUN = True

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_PCP_COMBINE_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH}.tm00_G212

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/PCPCombine/PCPCombine_user_
↪defined
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = wrfprs_ruc13_{init?fmt=%Y%m%d%H}_f{lead?fmt=%HH}_A{level?
↪fmt=%HH}.nc


###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

FCST_PCP_COMBINE_METHOD = USER_DEFINED

FCST_PCP_COMBINE_COMMAND = -derive sum,min,max,range,mean,stdev,vld_count {FCST_PCP_COMBINE_
```

(continues on next page)

(continued from previous page)

```
↪INPUT_DIR}/{init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH}.tm00_G212 {FCST_PCP_COMBINE_
↪INPUT_DIR}/{init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH?shift=-3H}.tm00_G212 {FCST_PCP_
↪COMBINE_INPUT_DIR}/{init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH?shift=-6H}.tm00_G212
↪{FCST_PCP_COMBINE_INPUT_DIR}/{init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH?shift=-9H}.
↪tm00_G212 {FCST_PCP_COMBINE_INPUT_DIR}/{init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH?
↪shift=-12H}.tm00_G212 {FCST_PCP_COMBINE_INPUT_DIR}/{init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?
↪fmt=%HH?shift=-15H}.tm00_G212 -field 'name="{FCST_PCP_COMBINE_INPUT_NAMES}"; level="{FCST_
↪PCP_COMBINE_INPUT_LEVELS}";'

#LOG_PCP_COMBINE_VERBOSITY = 2

FCST_IS_PROB = false

FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB

FCST_PCP_COMBINE_INPUT_ACCUMS = 3H
FCST_PCP_COMBINE_INPUT_NAMES = APCP
FCST_PCP_COMBINE_INPUT_LEVELS = A03

FCST_PCP_COMBINE_OUTPUT_ACCUM = A24
```

MET Configuration

None. PCPCombine does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in PCPCombine_user_defined.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_user_defined.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PCPCombine_user_defined.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_user_defined.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases

- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/PCPCombine/PCPCombine_user_defined (relative to **OUTPUT_BASE**) and will contain the following files:

- wrfprs_ruc13_2005080700_f24_A24.nc

Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PCPCombine.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.17.6 PCPCombine: Custom String Looping Use Case

met_tool_wrapper/PCPCombine/PCPCombine_loop_custom.conf

Scientific Objective

None. This wrapper's purpose is to demonstrate the ability to read in a user-defined list of strings, processing each item in the list for the given run time.

Datasets

Forecast: WRF-ARW precipitation 24h accumulation fields

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 551) section for more information.

Data Source: WRF-AFW

METplus Components

This use case utilizes the METplus PCPCombine wrapper to run across a user-provided list of strings, executing each item in the list for each run time. In this example, the ADD mode of PCPCombine is used, but only a single file is processed for each run time. Because it is executed in this manner, the output will match the input.

METplus Workflow

PCPCombine is the only tool called in this example. It processes the following run times:

Valid: 2009-12-31_12Z
Forecast lead: 24 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PCPCombine/PCPCombine_loop_custom.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PCPCombine/PCPCombine_
↪loop_custom.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PcpCombine


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2009123112
INIT_END = 2009123112
INIT_INCREMENT = 1M

LEAD_SEQ = 24H

PCP_COMBINE_CUSTOM_LOOP_LIST = arw-fer-gep1, arw-fer-gep5, arw-sch-gep2, arw-sch-gep6, arw-
↪tom-gep3, arw-tom-gep7
```

(continues on next page)

(continued from previous page)

```
###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_PCP_COMBINE_RUN = True

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_PCP_COMBINE_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/{custom?fmt=%s}/d01_{init?fmt=%Y%m%d%H}
↪_0{lead?fmt=%HH}00.grib

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/PCPCombine/PCPCombine_loop_
↪custom
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = {custom?fmt=%s}/d01_{init?fmt=%Y%m%d%H}_0{lead?fmt=%HH}00.
↪nc


###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

#LOG_PCP_COMBINE_VERBOSITY = 2

FCST_PCP_COMBINE_METHOD = ADD

FCST_PCP_COMBINE_CONSTANT_INIT = True

FCST_PCP_COMBINE_MAX_FORECAST = 2d

FCST_IS_PROB = false

FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB
FCST_PCP_COMBINE_INPUT_ACCUMS = 24H

FCST_PCP_COMBINE_OUTPUT_ACCUM = 24H
FCST_PCP_COMBINE_OUTPUT_NAME = APCP
```

MET Configuration

None. PCPCombine does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in PCPCombine_loop_custom.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_loop_custom.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PCPCombine_loop_custom.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_loop_custom.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/PCPCombine/PCPCombine_loop_custom (relative to **OUTPUT_BASE**) and will contain the following folders:

- arw-fer-gep1
- arw-fer-gep5
- arw-sch-gep2
- arw-sch-gep6
- arw-tom-gep3
- arw-tom-gep7

and each of the folders will contain a single file titled:

- d01_2009123112_02400.nc

Keywords

Note:

- PCPCombineToolUseCase
- CustomStringLoopingUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PCPCombine.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.17.7 PCPCombine: Bucket Interval Use Case

met_tool_wrapper/PCPCombine/PCPCombine_bucket.conf

Scientific Objective

Build a 15 hour precipitation accumulation field from varying accumulation fields.

Datasets

Forecast: GFS precipitation accumulation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 555) section for more information.

Data Source: GFS

METplus Components

This use case utilizes the METplus PCPCombine wrapper to search for files to build the desired accumulation for a given run time using a filename template and a list of available input accumulations. If enough files meeting the criteria are found to build the output accumulation, it will generate a command to run PCPCombine to combine the data.

METplus Workflow

PCPCombine is the only tool called in this example. It processes the following run times:

Valid: 2012-04-09_00Z
Forecast lead: 15 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PCPCombine/PCPCombine_bucket.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PCPCombine/PCPCombine_
↪bucket.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PcpCombine


###
```

(continues on next page)

(continued from previous page)

```
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2012040900
INIT_END = 2012040900
INIT_INCREMENT = 86400

LEAD_SEQ = 15H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_PCP_COMBINE_RUN = True

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/met_test/new/gfs
FCST_PCP_COMBINE_INPUT_TEMPLATE = gfs_{init?fmt=%Y%m%d%H}_F{lead?fmt=%3H}.grib

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/PCPCombine/PCPCombine_bucket
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = gfs_{valid?fmt=%Y%m%d%H}_A{level?fmt=%3H}.nc



###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

FCST_PCP_COMBINE_METHOD = ADD

#LOG_PCP_COMBINE_VERBOSITY = 2

FCST_PCP_COMBINE_MAX_FORECAST = 2d
```

(continues on next page)

(continued from previous page)

```
FCST_IS_PROB = false

FCST_PCP_COMBINE_BUCKET_INTERVAL = 6H
FCST_PCP_COMBINE_INPUT_ACCUMS = {lead}

FCST_PCP_COMBINE_OUTPUT_ACCUM = 15H
FCST_PCP_COMBINE_OUTPUT_NAME = APCP
```

MET Configuration

None. PCPCombine does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in PCPCombine_bucket.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_bucket.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PCPCombine_bucket.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_bucket.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/PCPCombine/PCPCombine_bucket (relative to **OUTPUT_BASE**) and will contain the following files:

- gfs_2012040915_A015.nc

Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PCPCombine.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.17.8 PCPCombine: ADD Use Case

met_tool_wrapper/PCPCombine/PCPCombine_add.conf

Scientific Objective

Build a 15 minute precipitation accumulation field from 5 minute accumulation fields.

Datasets

Forecast: NEWSe 5 minute precipitation accumulation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 559) section for more information.

Data Source: NEWSe

METplus Components

This use case utilizes the METplus PCPCombine wrapper to search for files to build the desired accumulation for a given run time using a filename template and a list of available input accumulations. If enough files meeting the criteria are found to build the output accumulation, it will generate a command to run PCPCombine to combine the data.

METplus Workflow

PCPCombine is the only tool called in this example. It processes the following run times:

Valid: 2019-08-02_18:15Z
Forecast lead: 15 minute

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PCPCombine/PCPCombine_add.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PCPCombine/PCPCombine_
↪add.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PCPCombine


###
```

(continues on next page)

(continued from previous page)

```
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H%M
VALID_BEG = 201908021815
VALID_END = 201908021815
VALID_INCREMENT = 1M

LEAD_SEQ = 15M


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_PCP_COMBINE_RUN = True

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/met_test/new
FCST_PCP_COMBINE_INPUT_TEMPLATE = NEWSe_{init?fmt=%Y%m%d}_i{init?fmt=%H%M}_m0_f{valid?fmt=%H
↪%M}.nc

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/PCPCombine/PCPCombine_add
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = NEWSe5min_mem00_lag00.nc


###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

#LOG_PCP_COMBINE_VERBOSITY = 2

FCST_PCP_COMBINE_METHOD = ADD
```

(continues on next page)

(continued from previous page)

```
FCST_PCP_COMBINE_MAX_FORECAST = 2d
FCST_PCP_COMBINE_CONSTANT_INIT = FALSE

FCST_IS_PROB = false

FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB

FCST_PCP_COMBINE_INPUT_ACCUMS = 5M
FCST_PCP_COMBINE_INPUT_NAMES = A000500
FCST_PCP_COMBINE_INPUT_LEVELS = Surface

FCST_PCP_COMBINE_OUTPUT_ACCUM = 15M
FCST_PCP_COMBINE_OUTPUT_NAME = A001500
```

MET Configuration

None. PCPCombine does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in PCPCombine_add.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_add.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PCPCombine_add.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PCPCombine/
↪PCPCombine_add.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/PCPCombine/PCPCombine_add (relative to **OUTPUT_BASE**) and will contain the following files:

- NEWSe5min_mem00_lag00.nc

Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PCPCombine.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.18 PlotDataPlane

7.1.32.18.1 PlotDataPlane: GRIB1 Input

met_tool_wrapper/PlotDataPlane/PlotDataPlane_grib1.conf

Scientific Objective

Generate a postscript image to test if the input data can be read by the MET tools.

Datasets

Input: Sample GRIB1 file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

Data Source: NAM

METplus Components

This use case utilizes the METplus PlotDataPlane wrapper to generate a command to run the MET tool PlotDataPlane if all required files are found.

METplus Workflow

PlotDataPlane is the only tool called in this example. It processes the following run time:

Valid: 2007-03-30 0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PlotDataPlane/PlotDataPlane_grib1.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PlotDataPlane/
```

(continues on next page)

(continued from previous page)

```
↪PlotDataPlane_grib1.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PlotDataPlane


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20070330
VALID_END = 20070330
VALID_INCREMENT = 1M

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

PLOT_DATA_PLANE_INPUT_DIR =
PLOT_DATA_PLANE_INPUT_TEMPLATE = {INPUT_BASE}/met_test/data/sample_fcst/{valid?fmt=%Y%m%d%H}/
↪nam.t{valid?fmt=%H}z.awip1236.tm00.{valid?fmt=%Y%m%d}.grb

PLOT_DATA_PLANE_OUTPUT_DIR =
```

(continues on next page)

(continued from previous page)

```
PLOT_DATA_PLANE_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/plot_data_plane/nam.t00z.
↪awip1236.tm{valid?fmt=%H}.{valid?fmt=%Y%m%d}_TMPZ2.ps


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

PLOT_DATA_PLANE_FIELD_NAME = TMP
PLOT_DATA_PLANE_FIELD_LEVEL = Z2


###
# PlotDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#plotdataplane
###

LOG_PLOT_DATA_PLANE_VERBOSITY = 1

PLOT_DATA_PLANE_TITLE = GRIB1 NAM {PLOT_DATA_PLANE_FIELD_LEVEL} {PLOT_DATA_PLANE_FIELD_NAME}

PLOT_DATA_PLANE_COLOR_TABLE =

PLOT_DATA_PLANE_RANGE_MIN_MAX =
```

MET Configuration

This tool does not use a MET configuration file.

Running METplus

This use case can be run two ways:

1) Passing in PlotDataPlane_grib1.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PlotDataPlane/
↪PlotDataPlane_grib1.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PlotDataPlane_grib1.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PlotDataPlane/
↪PlotDataPlane_grib1.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/plot_data_plane (relative to **OUTPUT_BASE**) and will contain the following file:

- nam.t00z.awip1236.tm00.20070330_TMPZ2.ps

Keywords

Note:

- PlotDataPlaneToolUseCase
- GRIBFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PlotDataPlane.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.18.2 PlotDataPlane: NetCDF Input

met_tool_wrapper/PlotDataPlane/PlotDataPlane_netcdf.conf

Scientific Objective

Generate a postscript image to test if the input data can be read by the MET tools.

Datasets

Input: Sample NetCDF file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus PlotDataPlane wrapper to generate a command to run the MET tool PlotDataPlane if all required files are found.

METplus Workflow

PlotDataPlane is the only tool called in this example. It processes the following run time:

Valid: 2007-03-30 0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PlotDataPlane/PlotDataPlane_netcdf.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PlotDataPlane/
↪PlotDataPlane_netcdf.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PlotDataPlane


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2005080712
VALID_END = 2005080712
VALID_INCREMENT = 1M

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

PLOT_DATA_PLANE_INPUT_DIR =
PLOT_DATA_PLANE_INPUT_TEMPLATE = {INPUT_BASE}/met_test/out/pcp_combine/sample_fcst_12L_
↪{valid?fmt=%Y%m%d%H}V_12A.nc

PLOT_DATA_PLANE_OUTPUT_DIR =
PLOT_DATA_PLANE_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/plot_data_plane/sample_fcst_
↪12L_{valid?fmt=%Y%m%d%H}V_12A_APCP12_NC_MET.ps


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

PLOT_DATA_PLANE_FIELD_NAME = APCP_12
PLOT_DATA_PLANE_FIELD_LEVEL = "(*,*)"


###
# PlotDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#plotdataplane
###

LOG_PLOT_DATA_PLANE_VERBOSITY = 1

PLOT_DATA_PLANE_TITLE = NC MET 12-hour APCP

PLOT_DATA_PLANE_COLOR_TABLE =

PLOT_DATA_PLANE_RANGE_MIN_MAX = 1.0 3.0
```

MET Configuration

This tool does not use a MET configuration file.

Running METplus

This use case can be run two ways:

1) Passing in PlotDataPlane_netcdf.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PlotDataPlane/
↪PlotDataPlane_netcdf.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PlotDataPlane_netcdf.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PlotDataPlane/
↪PlotDataPlane_netcdf.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/plot_data_plane (relative to **OUTPUT_BASE**) and will contain the following file:

- sample_fcst_12L_2005080712V_12A_APCP12_NC_MET.ps

Keywords

Note:

- PlotDataPlaneToolUseCase
- NetCDFFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PlotDataPlane.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.18.3 PlotDataPlane: Python Embedding Input

met_tool_wrapper/PlotDataPlane/PlotDataPlane_python_embedding.conf

Scientific Objective

Generate a postscript image to test if the input data can be read by the MET tools.

Datasets

Input: Sample Python Embedding script/file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus PlotDataPlane wrapper to generate a command to run the MET tool PlotDataPlane if all required files are found.

METplus Workflow

PlotDataPlane is the only tool called in this example. It processes the following run time:

Valid: 2005-08-07 12Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PlotDataPlane/PlotDataPlane_python_embedding.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PlotDataPlane/
↪PlotDataPlane_python_embedding.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PlotDataPlane


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2005080712
VALID_END = 2005080712
VALID_INCREMENT = 1M

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

PLOT_DATA_PLANE_INPUT_DIR =
PLOT_DATA_PLANE_INPUT_TEMPLATE = PYTHON_NUMPY

PLOT_DATA_PLANE_OUTPUT_DIR =
PLOT_DATA_PLANE_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/plot_data_plane/py_embed_
↪fcst.ps


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

PLOT_DATA_PLANE_FIELD_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py
↪{INPUT_BASE}/met_test/data/python/fcst.txt FCST


###
# PlotDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#plotdataplane
###

# Verbosity of MET output - overrides LOG_VERBOSITY for PlotDataPlane only
LOG_PLOT_DATA_PLANE_VERBOSITY = 1

PLOT_DATA_PLANE_TITLE = Python Embedding FCST
```

(continues on next page)

(continued from previous page)

```
PLOT_DATA_PLANE_COLOR_TABLE =

PLOT_DATA_PLANE_RANGE_MIN_MAX =
```

MET Configuration

This tool does not use a MET configuration file.

Python Embedding

This use case calls a Python script to read the input data. The Python script is stored in the MET repository: /path/to/MET/installation/share/met/python/read_ascii_numpy.py

read_ascii_numpy.py

Running METplus

This use case can be run two ways:

1) Passing in PlotDataPlane_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PlotDataPlane/
↪PlotDataPlane_python_embedding.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PlotDataPlane_python_embedding.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PlotDataPlane/
↪PlotDataPlane_python_embedding.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/plot_data_plane (relative to **OUTPUT_BASE**) and will contain the following file:

- py_embed_fcst.ps

Keywords

Note:

- PlotDataPlaneToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PlotDataPlane.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.19 PlotPointObs

7.1.32.19.1 PlotPointObs: Basic Use Case

met_tool_wrapper/PlotPointObs/PlotPointObs.conf

Scientific Objective

Generate a postscript image to plot point observations using a gridded file to define the domain to plot the points.

Datasets

Point Observations: NetCDF generated by PB2NC and ASCII2NC
Grid: GRIB2 NAM

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

Data Source: NAM and NDAS

METplus Components

This use case utilizes the METplus PlotPointObs wrapper to generate a command to run the MET tool plot_point_obs if all required files are found.

METplus Workflow

PlotPointObs is the only tool called in this example. It processes the following run time:

Valid: 2012-04-09 12Z
Forecast lead: 12 hour

METplus Configuration

METplus first loads the default configuration file found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line: parm/use_cases/met_tool_wrapper/PlotPointObs/PlotPointObs.conf

```
[config]


# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PlotPointObs/
↪PlotPointObs.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PlotPointObs


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

PLOT_POINT_OBS_RUNTIME_FREQ = RUN_ONCE_FOR_EACH

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2012040912
VALID_END = 2012040912
VALID_INCREMENT = 1M

LEAD_SEQ = 12H
```

(continues on next page)

(continued from previous page)

```
###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

PLOT_POINT_OBS_INPUT_DIR = {INPUT_BASE}/met_test/new/test_out
PLOT_POINT_OBS_INPUT_TEMPLATE =
   pb2nc/ndas.{valid?fmt=%Y%m%d}.t{valid?fmt=%H}z.prepbufr.tm00.nc,
   ascii2nc/trmm_{valid?fmt=%Y%m%d%H}_3hr.nc

PLOT_POINT_OBS_GRID_INPUT_DIR = {INPUT_BASE}/met_test/new/model_data/grib2/nam
PLOT_POINT_OBS_GRID_INPUT_TEMPLATE = nam_{init?fmt=%Y%m%d%H}_F{lead?fmt=%3H}.grib2

PLOT_POINT_OBS_OUTPUT_DIR = {OUTPUT_BASE}/plot_point_obs
PLOT_POINT_OBS_OUTPUT_TEMPLATE = nam_and_ndas.{valid?fmt=%Y%m%d}.t{valid?fmt=%H}z.prepbufr_
↪CONFIG.ps

#PLOT_POINT_OBS_SKIP_IF_OUTPUT_EXISTS = False


###
# PlotPointObs Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#plotpointobs
###

PLOT_POINT_OBS_TITLE = NAM {init?fmt=%Y%m%d%H} F{lead?fmt=%2H} vs NDAS 500mb RH and TRMM 3h >
↪ 0

#PLOT_POINT_OBS_CONFIG_FILE = {PARM_BASE}/met_config/PlotPointObsConfig_wrapped

#LOG_PLOT_POINT_OBS_VERBOSITY = 2

PLOT_POINT_OBS_GRID_DATA_FIELD = { name = "RH"; level = "P500"; }
PLOT_POINT_OBS_GRID_DATA_REGRID_TO_GRID = NONE
#PLOT_POINT_OBS_GRID_DATA_REGRID_METHOD =
#PLOT_POINT_OBS_GRID_DATA_REGRID_WIDTH =
#PLOT_POINT_OBS_GRID_DATA_REGRID_VLD_THRESH =
#PLOT_POINT_OBS_GRID_DATA_REGRID_SHAPE =
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_COLOR_TABLE = MET_BASE/colortables/NCL_colortables/
↪BlueGreen14.ctable
#PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_PLOT_MIN =
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_PLOT_MAX = 100.0
#PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_COLORBAR_FLAG =
```

(continues on next page)

(continued from previous page)

```
#PLOT_POINT_OBS_MSG_TYP =
#PLOT_POINT_OBS_SID_INC =
#PLOT_POINT_OBS_SID_EXC =
#PLOT_POINT_OBS_OBS_VAR =
#PLOT_POINT_OBS_OBS_GC =
#PLOT_POINT_OBS_OBS_QUALITY =
#PLOT_POINT_OBS_VALID_BEG =
#PLOT_POINT_OBS_VALID_END =
#PLOT_POINT_OBS_LAT_THRESH =
#PLOT_POINT_OBS_LON_THRESH =
#PLOT_POINT_OBS_ELV_THRESH =
#PLOT_POINT_OBS_HGT_THRESH =
#PLOT_POINT_OBS_PRS_THRESH =
#PLOT_POINT_OBS_OBS_THRESH =
#PLOT_POINT_OBS_CENSOR_THRESH =
#PLOT_POINT_OBS_CENSOR_VAL =
#PLOT_POINT_OBS_DOTSIZE =
#PLOT_POINT_OBS_LINE_COLOR =
#PLOT_POINT_OBS_LINE_WIDTH =
#PLOT_POINT_OBS_FILL_COLOR =

#PLOT_POINT_OBS_FILL_PLOT_INFO_FLAG =
#PLOT_POINT_OBS_FILL_PLOT_INFO_COLOR_TABLE =
#PLOT_POINT_OBS_FILL_PLOT_INFO_PLOT_MIN =
#PLOT_POINT_OBS_FILL_PLOT_INFO_PLOT_MAX =
#PLOT_POINT_OBS_FILL_PLOT_INFO_COLORBAR_FLAG =

PLOT_POINT_OBS_POINT_DATA =
  {
    msg_typ = "ADPSFC";
    obs_gc = 61;
    obs_thresh = > 0.0;
    fill_color = [0, 0, 255];
  },
  {
    msg_typ = "ADPSFC";
    obs_var = "RH";
    fill_color = [100, 100, 100];
  },
  {
    msg_typ = "ADPUPA";
    obs_var = "RH";
    prs_thresh = == 500;
    dotsize(x) = 7.5;
    line_color = [0, 0, 0];
```

(continues on next page)

(continued from previous page)

```
    fill_plot_info = {
      flag = TRUE;
      color_table = "MET_BASE/colortables/NCL_colortables/BlueGreen14.ctable";
      plot_min = 0.0;
      plot_max = 100.0;
      colorbar_flag = FALSE;
    }
  }
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *PlotPointObs MET Configuration* (page 208) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Plot-Point-Obs configuration file.
//
// For additional information, please see the MET Users Guide.
//
////////////////////////////////////////////////////////////////////////////////

// Gridded data plotting options

//grid_data = {
${METPLUS_GRID_DATA_DICT}

////////////////////////////////////////////////////////////////////////////////

// Point data filtering options
// May be set separately in each "point_data" entry

//msg_typ =
${METPLUS_MSG_TYP}
```

(continues on next page)

(continued from previous page)

```
//sid_inc =
${METPLUS_SID_INC}

//sid_exc =
${METPLUS_SID_EXC}

//obs_var =
${METPLUS_OBS_VAR}

//obs_gc =
${METPLUS_OBS_GC}

//obs_quality =
${METPLUS_OBS_QUALITY}

//valid_beg =
${METPLUS_VALID_BEG}

//valid_end =
${METPLUS_VALID_END}

//lat_thresh =
${METPLUS_LAT_THRESH}

//lon_thresh =
${METPLUS_LON_THRESH}

//elv_thresh =
${METPLUS_ELV_THRESH}

//hgt_thresh =
${METPLUS_HGT_THRESH}

//prs_thresh =
${METPLUS_PRS_THRESH}

//obs_thresh =
${METPLUS_OBS_THRESH}

// Point data pre-processing options
// May be set separately in each "point_data" entry

//convert(x)    = x;

//censor_thresh =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CENSOR_THRESH}

//censor_val =
${METPLUS_CENSOR_VAL}


// Point data plotting options
// May be set separately in each "point_data" entry

//dotsize =
${METPLUS_DOTSIZE}

//line_color =
${METPLUS_LINE_COLOR}

//line_width =
${METPLUS_LINE_WIDTH}

//fill_color =
${METPLUS_FILL_COLOR}

//fill_plot_info = {
${METPLUS_FILL_PLOT_INFO_DICT}

// Array of point data filtering, pre-processing, and plotting options
//point_data =
${METPLUS_POINT_DATA}

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

Pass the use case configuration file to the run_metplus.py script along with any user-specific system configuration files if desired:

```
run_metplus.py /path/to/METplus/parm/use_cases/met_tool_wrapper/PlotPointObs/PlotPointObs.
↪conf /path/to/user_system.conf
```

See *Running METplus* (page 26) for more information.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in plot_point_obs (relative to **OUTPUT_BASE**) and will contain the following file:

- nam_and_ndas.20120409.t12z.prepbufr_CONFIG.ps

Keywords

Note:

- PlotPointObsToolUseCase
- GRIBFileUseCase
- NetCDFFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PlotPointObs.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.20 Point2Grid

7.1.32.20.1 Point2Grid: Basic Use Case

met_tool_wrapper/Point2Grid/Point2Grid.conf

Scientific Objective

Putting point observations onto a grid for use with other tools.

Datasets

Observations: Stage 2 NetCDF 1-hour Precipitation Accumulation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 584) section for more information.

METplus Components

This use case utilizes the METplus Point2Grid wrapper to generate a command to run the MET tool Point2Grid if all required files are found.

METplus Workflow

Point2Grid is the only tool called in this example. It processes the following run time:

Init: 2017-06-01_0Z

This use case puts point observations onto a specified grid

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/Point2Grid/Point2Grid.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/Point2Grid/Point2Grid.
↪html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = Point2Grid


###
```

(continues on next page)

(continued from previous page)

```
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2017060100
INIT_END = 2017060300
INIT_INCREMENT = 24H

LEAD_SEQ = 12H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

POINT2GRID_INPUT_DIR =
POINT2GRID_INPUT_TEMPLATE = {INPUT_BASE}/met_test/data/sample_obs/prepbufr/sample_pb.nc

POINT2GRID_OUTPUT_DIR =
POINT2GRID_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/Point2Grid/grid.{init?fmt=%Y%d%H}
↪.nc

POINT2GRID_FILE_WINDOW_BEGIN = 0
POINT2GRID_FILE_WINDOW_END = 0


###
# Point2Grid Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#point2grid
###

# LOG_POINT2GRID_VERBOSITY = 1

POINT2GRID_REGRID_TO_GRID = G212
```

(continues on next page)

(continued from previous page)

```
POINT2GRID_REGRID_METHOD = MAX
#POINT2GRID_REGRID_WIDTH =
#POINT2GRID_REGRID_VLD_THRESH =
#POINT2GRID_REGRID_SHAPE =
#POINT2GRID_REGRID_CONVERT =
#POINT2GRID_REGRID_CENSOR_THRESH =
#POINT2GRID_REGRID_CENSOR_VAL =

POINT2GRID_INPUT_FIELD =TMP
POINT2GRID_INPUT_LEVEL =

#POINT2GRID_QC_FLAGS =

POINT2GRID_ADP =


POINT2GRID_GAUSSIAN_DX = 81.271
POINT2GRID_GAUSSIAN_RADIUS = 120

POINT2GRID_PROB_CAT_THRESH =

POINT2GRID_VLD_THRESH =
```

MET Configuration

None. Point2Grid does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in Point2Grid.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/Point2Grid/
↪Point2Grid.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in Point2Grid.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/Point2Grid/
↪Point2Grid.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in point2grid (relative to **OUTPUT_BASE**) and will contain the following files:

- grid.20170100.nc
- grid.20170200.nc
- grid.20170300.nc

Keywords

Note:

- Point2GridToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-Point2Grid.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.21 PointStat

7.1.32.21.1 PointStat: Using Python Embedding for Point Observations

met_tool_wrapper/PointStat/PointStat_python_embedding_obs.conf

Scientific Objective

Compare hourly forecasts for temperature, u-, and v-wind components to observations in a 3-hour observation window. Generate statistics of the results. Calls a Python Embedding script to read point observations.

Datasets

Forecast: NAM temperature, u-wind component, and v-wind component
Observation: prepBUFR data that has been converted to NetCDF format via PB2NC

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 595) section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus PointStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool point_stat if all required files are found. This example demonstrates how to configure a use case to call a Python Embedding script to read in point observations into point_stat.

METplus Workflow

PointStat is the only tool called in this example. It processes the following run times:

Valid: 2007-03-30_0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PointStat/PointStat_python_embedding_obs.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PointStat/PointStat_
↪python_embedding_obs.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PointStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20070330
INIT_END = 20070330
INIT_INCREMENT = 1M

LEAD_SEQ = 36


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_POINT_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/nam.t00z.awip1236.tm00.{init?fmt=%Y%m%d}
↪.grb

OBS_POINT_STAT_INPUT_DIR =
OBS_POINT_STAT_INPUT_TEMPLATE = PYTHON_NUMPY= {MET_INSTALL_DIR}/share/met/python/examples/
↪read_met_point_obs.py {INPUT_BASE}/met_test/out/pb2nc/sample_pb.nc

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/point_stat_py_embed_obs

POINT_STAT_CLIMO_MEAN_INPUT_DIR =
POINT_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

POINT_STAT_CLIMO_STDEV_INPUT_DIR =
POINT_STAT_CLIMO_STDEV_INPUT_TEMPLATE =


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

POINT_STAT_ONCE_PER_FIELD = False

#POINT_STAT_FCST_FILE_TYPE =
#POINT_STAT_OBS_FILE_TYPE =

FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = P750-900
FCST_VAR1_THRESH = <=273, >273
OBS_VAR1_NAME = TMP
OBS_VAR1_LEVELS = P750-900
OBS_VAR1_THRESH = <=273, >273
OBS_VAR1_OPTIONS = set_attr_units = "K"

FCST_VAR2_NAME = UGRD
FCST_VAR2_LEVELS = Z10
FCST_VAR2_THRESH = >=5
OBS_VAR2_NAME = UGRD
OBS_VAR2_LEVELS = Z10
OBS_VAR2_THRESH = >=5
OBS_VAR2_OPTIONS = set_attr_units = "m/s"
```

(continues on next page)

(continued from previous page)

```
FCST_VAR3_NAME = VGRD
FCST_VAR3_LEVELS = Z10
FCST_VAR3_THRESH = >=5
OBS_VAR3_NAME = VGRD
OBS_VAR3_LEVELS = Z10
OBS_VAR3_THRESH = >=5
OBS_VAR3_OPTIONS = set_attr_units = "m/s"


###
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###

#LOG_POINT_STAT_VERBOSITY = 2

POINT_STAT_CONFIG_FILE ={PARM_BASE}/met_config/PointStatConfig_wrapped

#POINT_STAT_OBS_QUALITY_INC = 1, 2, 3
#POINT_STAT_OBS_QUALITY_EXC =

POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST
#POINT_STAT_CLIMO_STDEV_TIME_INTERP_METHOD =

#POINT_STAT_INTERP_VLD_THRESH =
#POINT_STAT_INTERP_SHAPE =
POINT_STAT_INTERP_TYPE_METHOD = BILIN
POINT_STAT_INTERP_TYPE_WIDTH = 2

#POINT_STAT_OUTPUT_FLAG_FHO =
#POINT_STAT_OUTPUT_FLAG_CTC =
#POINT_STAT_OUTPUT_FLAG_CTS =
#POINT_STAT_OUTPUT_FLAG_MCTC =
#POINT_STAT_OUTPUT_FLAG_MCTS =
#POINT_STAT_OUTPUT_FLAG_CNT =
POINT_STAT_OUTPUT_FLAG_SL1L2 = STAT
#POINT_STAT_OUTPUT_FLAG_SAL1L2 =
POINT_STAT_OUTPUT_FLAG_VL1L2 = STAT
#POINT_STAT_OUTPUT_FLAG_VAL1L2 =
#POINT_STAT_OUTPUT_FLAG_VCNT =
#POINT_STAT_OUTPUT_FLAG_PCT =
#POINT_STAT_OUTPUT_FLAG_PSTD =
#POINT_STAT_OUTPUT_FLAG_PJC =
#POINT_STAT_OUTPUT_FLAG_PRC =
```

(continues on next page)

(continued from previous page)

```
#POINT_STAT_OUTPUT_FLAG_ECNT =
#POINT_STAT_OUTPUT_FLAG_RPS =
#POINT_STAT_OUTPUT_FLAG_ECLV =
#POINT_STAT_OUTPUT_FLAG_MPR =
#POINT_STAT_OUTPUT_FLAG_ORANK =

#POINT_STAT_CLIMO_CDF_BINS = 1
#POINT_STAT_CLIMO_CDF_CENTER_BINS = False
#POINT_STAT_CLIMO_CDF_WRITE_BINS = True
#POINT_STAT_CLIMO_CDF_DIRECT_PROB =

#POINT_STAT_HSS_EC_VALUE =

OBS_POINT_STAT_WINDOW_BEGIN = -5400
OBS_POINT_STAT_WINDOW_END = 5400

POINT_STAT_OFFSETS = 0

MODEL = WRF

POINT_STAT_DESC = NA
OBTYPE =

POINT_STAT_REGRID_TO_GRID = NONE
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2

POINT_STAT_OUTPUT_PREFIX =

#POINT_STAT_OBS_VALID_BEG = {valid?fmt=%Y%m%d_%H}
#POINT_STAT_OBS_VALID_END = {valid?fmt=%Y%m%d_%H}

POINT_STAT_MASK_GRID = DTC165, DTC166
POINT_STAT_MASK_POLY = MET_BASE/poly/LMV.poly
POINT_STAT_MASK_SID =
#POINT_STAT_MASK_LLPNT =

POINT_STAT_MESSAGE_TYPE = ADPUPA, ADPSFC

#POINT_STAT_HIRA_FLAG =
#POINT_STAT_HIRA_WIDTH =
#POINT_STAT_HIRA_VLD_THRESH =
#POINT_STAT_HIRA_COV_THRESH =
#POINT_STAT_HIRA_SHAPE =
#POINT_STAT_HIRA_PROB_CAT_THRESH =
```

(continues on next page)

(continued from previous page)

```
#POINT_STAT_MESSAGE_TYPE_GROUP_MAP =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *PointStat MET Configuration* (page 218) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in PointStat_python_embedding_obs.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PointStat/PointStat_
↪python_embedding_obs.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PointStat_python_embedding_obs.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PointStat/PointStat_
↪python_embedding_obs.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in point_stat_py_embed_obs (relative to **OUTPUT_BASE**) and will contain the following files:

- point_stat_360000L_20070331_120000V.stat

Keywords

Note:

- PointStatToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PointStat.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.21.2 PointStat: Basic Use Case

met_tool_wrapper/PointStat/PointStat.conf

Scientific Objective

Compare hourly forecasts for temperature, u-, and v-wind components to observations in a 3-hour observation window. Generate statistics of the results.

Datasets

Forecast: NAM temperature, u-wind component, and v-wind component
Observation: prepBUFR data that has been converted to NetCDF format via PB2NC

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 606) section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus PointStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool point_stat if all required files are found.

METplus Workflow

PointStat is the only tool called in this example. It processes the following run times:

Valid: 2007-03-30_0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PointStat/PointStat.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PointStat/PointStat.
↪html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PointStat
```

(continues on next page)

(continued from previous page)

```
###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20070330
INIT_END = 20070330
INIT_INCREMENT = 1M

LEAD_SEQ = 36


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_POINT_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/nam.t00z.awip1236.tm00.{init?fmt=%Y%m%d}
↪.grb

OBS_POINT_STAT_INPUT_DIR = {INPUT_BASE}/met_test/out/pb2nc
OBS_POINT_STAT_INPUT_TEMPLATE = sample_pb.nc

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/point_stat

POINT_STAT_CLIMO_MEAN_INPUT_DIR =
POINT_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

POINT_STAT_CLIMO_STDEV_INPUT_DIR =
POINT_STAT_CLIMO_STDEV_INPUT_TEMPLATE =


###
# Field Info
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

POINT_STAT_ONCE_PER_FIELD = False

#POINT_STAT_FCST_FILE_TYPE =
#POINT_STAT_OBS_FILE_TYPE =

FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = P750-900
FCST_VAR1_THRESH = <=273, >273
OBS_VAR1_NAME = TMP
OBS_VAR1_LEVELS = P750-900
OBS_VAR1_THRESH = <=273, >273

FCST_VAR2_NAME = UGRD
FCST_VAR2_LEVELS = Z10
FCST_VAR2_THRESH = >=5
OBS_VAR2_NAME = UGRD
OBS_VAR2_LEVELS = Z10
OBS_VAR2_THRESH = >=5

FCST_VAR3_NAME = VGRD
FCST_VAR3_LEVELS = Z10
FCST_VAR3_THRESH = >=5
OBS_VAR3_NAME = VGRD
OBS_VAR3_LEVELS = Z10
OBS_VAR3_THRESH = >=5


###
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###

#LOG_POINT_STAT_VERBOSITY = 2

POINT_STAT_CONFIG_FILE ={PARM_BASE}/met_config/PointStatConfig_wrapped

#POINT_STAT_OBS_QUALITY_INC = 1, 2, 3
#POINT_STAT_OBS_QUALITY_EXC =

POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST
#POINT_STAT_CLIMO_STDEV_TIME_INTERP_METHOD =
```

(continues on next page)

(continued from previous page)

```
#POINT_STAT_INTERP_VLD_THRESH =
#POINT_STAT_INTERP_SHAPE =
POINT_STAT_INTERP_TYPE_METHOD = BILIN
POINT_STAT_INTERP_TYPE_WIDTH = 2

#POINT_STAT_OUTPUT_FLAG_FHO =
#POINT_STAT_OUTPUT_FLAG_CTC =
#POINT_STAT_OUTPUT_FLAG_CTS =
#POINT_STAT_OUTPUT_FLAG_MCTC =
#POINT_STAT_OUTPUT_FLAG_MCTS =
#POINT_STAT_OUTPUT_FLAG_CNT =
POINT_STAT_OUTPUT_FLAG_SL1L2 = STAT
#POINT_STAT_OUTPUT_FLAG_SAL1L2 =
POINT_STAT_OUTPUT_FLAG_VL1L2 = STAT
#POINT_STAT_OUTPUT_FLAG_VAL1L2 =
#POINT_STAT_OUTPUT_FLAG_VCNT =
#POINT_STAT_OUTPUT_FLAG_PCT =
#POINT_STAT_OUTPUT_FLAG_PSTD =
#POINT_STAT_OUTPUT_FLAG_PJC =
#POINT_STAT_OUTPUT_FLAG_PRC =
#POINT_STAT_OUTPUT_FLAG_ECNT =
#POINT_STAT_OUTPUT_FLAG_ORANK =
#POINT_STAT_OUTPUT_FLAG_RPS =
#POINT_STAT_OUTPUT_FLAG_ECLV =
#POINT_STAT_OUTPUT_FLAG_MPR =
#POINT_STAT_OUTPUT_FLAG_SEEPS =
#POINT_STAT_OUTPUT_FLAG_SEEPS_MPR =

#POINT_STAT_CLIMO_CDF_BINS = 1
#POINT_STAT_CLIMO_CDF_CENTER_BINS = False
#POINT_STAT_CLIMO_CDF_WRITE_BINS = True
#POINT_STAT_CLIMO_CDF_DIRECT_PROB =

#POINT_STAT_HSS_EC_VALUE =

#POINT_STAT_SEEPS_P1_THRESH =

OBS_POINT_STAT_WINDOW_BEGIN = -5400
OBS_POINT_STAT_WINDOW_END = 5400

POINT_STAT_OFFSETS = 0

MODEL = WRF

POINT_STAT_DESC = NA
```

(continues on next page)

(continued from previous page)

```
OBTYPE =

POINT_STAT_REGRID_TO_GRID = NONE
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2
#POINT_STAT_REGRID_VLD_THRESH =
#POINT_STAT_REGRID_SHAPE =
#POINT_STAT_REGRID_CONVERT =
#POINT_STAT_REGRID_CENSOR_THRESH =
#POINT_STAT_REGRID_CENSOR_VAL =

POINT_STAT_OUTPUT_PREFIX =

#POINT_STAT_OBS_VALID_BEG = {valid?fmt=%Y%m%d_%H}
#POINT_STAT_OBS_VALID_END = {valid?fmt=%Y%m%d_%H}

POINT_STAT_MASK_GRID = DTC165, DTC166
POINT_STAT_MASK_POLY = MET_BASE/poly/LMV.poly
POINT_STAT_MASK_SID =
#POINT_STAT_MASK_LLPNT =

POINT_STAT_MESSAGE_TYPE = ADPUPA, ADPSFC

#POINT_STAT_HIRA_FLAG =
#POINT_STAT_HIRA_WIDTH =
#POINT_STAT_HIRA_VLD_THRESH =
#POINT_STAT_HIRA_COV_THRESH =
#POINT_STAT_HIRA_SHAPE =
#POINT_STAT_HIRA_PROB_CAT_THRESH =

#POINT_STAT_MESSAGE_TYPE_GROUP_MAP =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *PointStat MET Configuration* (page 218) section of the User's Guide for more information on

the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
```

(continues on next page)

(continued from previous page)

```
rank_corr_flag = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
```

(continues on next page)

(continued from previous page)

```
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in PointStat.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PointStat/PointStat.
↪conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PointStat.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PointStat/PointStat.
↪conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in point_stat (relative to **OUTPUT_BASE**) and will contain the following files:

- point_stat_360000L_20070331_120000V.stat

Keywords

Note:

- PointStatToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PointStat.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.21.3 PointStat: Once Per Field

met_tool_wrapper/PointStat/PointStat_once_per_field.conf

Scientific Objective

Compare 3 hour forecast precipitation accumulations to observations of 3 hour precipitation accumulation. Generate statistics of the results.

Datasets

Forecast: NAM temperature, u-wind component, and v-wind component
Observation: prepBURF data that has been converted to NetCDF format via PB2NC

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 615) section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus PointStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool point_stat if all required files are found. This use case processes each field name/level separately to generate output files for each. POINT_STAT_OUTPUT_PREFIX is used to control the names of the output fields, referencing {CURRENT_FCST_NAME} and {CURRENT_FCST_LEVEL} to get information for each field.

METplus Workflow

PointStat is the only tool called in this example. It processes the following run times:

Valid: 2007-03-30_0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PointStat/PointStat_once_per_field.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PointStat/PointStat_
↪once_per_field.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PointStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20070330
VALID_END = 20070330
VALID_INCREMENT = 1M

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_POINT_STAT_INPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}/nam.t00z.awip1236.tm00.{valid?fmt=%Y%m
↪%d}.grb

OBS_POINT_STAT_INPUT_DIR = {INPUT_BASE}/met_test/out/pb2nc
OBS_POINT_STAT_INPUT_TEMPLATE = sample_pb.nc

POINT_STAT_CLIMO_MEAN_INPUT_DIR =
POINT_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

POINT_STAT_CLIMO_STDEV_INPUT_DIR =
POINT_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/point_stat


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

POINT_STAT_ONCE_PER_FIELD = True
```

(continues on next page)

(continued from previous page)

```
FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = P750-900
FCST_VAR1_THRESH = <=273, >273
OBS_VAR1_NAME = TMP
OBS_VAR1_LEVELS = P750-900
OBS_VAR1_THRESH = <=273, >273

FCST_VAR2_NAME = UGRD
FCST_VAR2_LEVELS = Z10
FCST_VAR2_THRESH = >=5
OBS_VAR2_NAME = UGRD
OBS_VAR2_LEVELS = Z10
OBS_VAR2_THRESH = >=5

FCST_VAR3_NAME = VGRD
FCST_VAR3_LEVELS = Z10
FCST_VAR3_THRESH = >=5
OBS_VAR3_NAME = VGRD
OBS_VAR3_LEVELS = Z10
OBS_VAR3_THRESH = >=5


###
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###

#LOG_POINT_STAT_VERBOSITY = 2

POINT_STAT_CONFIG_FILE ={PARM_BASE}/met_config/PointStatConfig_wrapped

POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST

POINT_STAT_INTERP_TYPE_METHOD = BILIN
POINT_STAT_INTERP_TYPE_WIDTH = 2

POINT_STAT_OUTPUT_FLAG_SL1L2 = STAT
POINT_STAT_OUTPUT_FLAG_VL1L2 = STAT

OBS_POINT_STAT_WINDOW_BEGIN = -5400
OBS_POINT_STAT_WINDOW_END = 5400

POINT_STAT_OFFSETS = 0

MODEL = WRF
```

(continues on next page)

(continued from previous page)

```
OBTYPE =

#POINT_STAT_DESC =

POINT_STAT_REGRID_TO_GRID = NONE
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2

#POINT_STAT_OUTPUT_PREFIX = {fcst_name?fmt=%s}_{fcst_level?fmt=%s}
POINT_STAT_OUTPUT_PREFIX = {CURRENT_FCST_NAME}_{CURRENT_FCST_LEVEL}

#POINT_STAT_OBS_VALID_BEG = {valid?fmt=%Y%m%d_%H}
#POINT_STAT_OBS_VALID_END = {valid?fmt=%Y%m%d_%H}

POINT_STAT_GRID = DTC165, DTC166
POINT_STAT_POLY = MET_BASE/poly/LMV.poly

POINT_STAT_STATION_ID =

POINT_STAT_MESSAGE_TYPE = ADPUPA, ADPSFC
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *PointStat MET Configuration* (page 218) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
```

(continues on next page)

(continued from previous page)

```
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
```

(continues on next page)

(continued from previous page)

```
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in PointStat.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PointStat/PointStat_
↪once_per_field.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PointStat_once_per_field.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PointStat/PointStat_
↪once_per_field.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify

the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in point_stat (relative to **OUTPUT_BASE**) and will contain the following files:

- point_stat_TMP_P750-900_360000L_20070331_120000V.stat
- point_stat_UGRD_Z10_360000L_20070331_120000V.stat
- point_stat_VGRD_Z10_360000L_20070331_120000V.stat

Keywords

Note:

- PointStatToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PointStat.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.21.4 PointStat: Using Python Embedding

met_tool_wrapper/PointStat/PointStat_python_embedding.conf

Scientific Objective

Read forecast data using a Python embedding script. Compare with point observation data.

Datasets

Forecast: NRL binary data (v-wind component)
Observation: prepBUFR data that has been converted to NetCDF format via PB2NC

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus PointStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool point_stat if all required files are found. The forecast data is read into PointStat using a Python embedding script

METplus Workflow

PointStat is the only tool called in this example. It processes the following run times:

Valid: 2020-09-06 6Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PointStat/PointStat_python_embedding.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PointStat/PointStat_
↪python_embedding.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PointStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2020090606
VALID_END = 2020090606
VALID_INCREMENT = 1M

LEAD_SEQ = 0H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/met_test/new/point_stat_input/vvwind
FCST_POINT_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_POINT_STAT_INPUT_DIR = {INPUT_BASE}/met_test/new/point_stat_input/prepbufr
OBS_POINT_STAT_INPUT_TEMPLATE = gdas.{valid?fmt=%Y%m%d}.t{valid?fmt=%H}z.nc

POINT_STAT_CLIMO_MEAN_INPUT_DIR =
POINT_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

POINT_STAT_CLIMO_STDEV_INPUT_DIR =
POINT_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/point_stat_py_embed


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

POINT_STAT_ONCE_PER_FIELD = False

SCRIPT_DIR = {PARM_BASE}/use_cases/met_tool_wrapper/PointStat/PointStat_python_embedding
FCST_VAR1_NAME = {SCRIPT_DIR}/read_NRL_binary.py {FCST_POINT_STAT_INPUT_DIR}/vvwind_zht_0010.
↪0_0000.0_glob360x181_{init?fmt=%Y%m%d%H}_{lead?fmt=%4H}0000_fcstfld

OBS_VAR1_NAME = VGRD
OBS_VAR1_LEVELS = Z0


###
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###

#LOG_POINT_STAT_VERBOSITY = 2

POINT_STAT_CONFIG_FILE ={PARM_BASE}/met_config/PointStatConfig_wrapped

POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST

POINT_STAT_INTERP_TYPE_METHOD = BILIN
```

(continues on next page)

(continued from previous page)

```
POINT_STAT_INTERP_TYPE_WIDTH = 2

POINT_STAT_OUTPUT_FLAG_SL1L2 = STAT
POINT_STAT_OUTPUT_FLAG_VL1L2 = STAT

OBS_POINT_STAT_WINDOW_BEGIN = -5400
OBS_POINT_STAT_WINDOW_END = 5400

POINT_STAT_OFFSETS = 0

#MODEL =

#POINT_STAT_DESC = NA

#POINT_STAT_REGRID_TO_GRID =
#POINT_STAT_REGRID_METHOD =
#POINT_STAT_REGRID_WIDTH =

POINT_STAT_OUTPUT_PREFIX =

#POINT_STAT_OBS_VALID_BEG = {valid?fmt=%Y%m%d_%H}
#POINT_STAT_OBS_VALID_END = {valid?fmt=%Y%m%d_%H}

POINT_STAT_GRID = FULL

POINT_STAT_POLY =
POINT_STAT_STATION_ID =

POINT_STAT_MESSAGE_TYPE = ADPUPA
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *PointStat MET Configuration* (page 218) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;
```

(continues on next page)

(continued from previous page)

```
//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}
```

(continues on next page)

(continued from previous page)

```
//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case calls a Python script to read the binary input data. /parm/use_cases/met_tool_wrapper/PointStat/PointStat_python_embedding/read_NRL_binary.py

```
import os
import sys
import re
import numpy as np
import datetime as dt

# var_info values are tuples (units, long_name)
# Taken from synoptic_files.f, with some units SI standardized
# e.g. mb->hPa
# Some of the long_names are unknown to me, hopefully
# someone more knowledgeable will fill these in
var_info = {
      'airtmp': ('C','Air Temperature'),
      'geopht': ('gpm','Geopotential Height'),
      'uuwind': ('m/s','Zonal Wind'),
      'vvwind': ('m/s','Meridional Wind'),
      'wndspd': ('m/s','Wind Speed'),
      'vpress': ('hPa','Vapor Pressure'),
      'prch2o': ('kg/m**2','Unknown'),
      'slpres': ('hPa','Sea Level Pressure'),
      'grdwet': ('percent','Ground Wetness'),
      'prtend': ('cPa/s','Unknown'),
      'grdtmp': ('K','Ground Temperature'),
      'terrht': ('m','Terrain Height'),
      'totcls': ('percent','Unknown'),
      'lowcld': ('percent','Low Cloud'),
      'midcld': ('percent','Mid Cloud'),
      'hghcld': ('percent','High Cloud'),
      'cupflx': ('kg/m**2/s','Unknown'),
      'conpcp': ('cm','Unknown'),
      'sblpcp': ('cm','Unknown'),
      'trpres': ('hPa','Terrain Pressure'),
      'snowdp': ('cm','Snow Depth'),
      'icecon': ('percent','Ice Concentration'),
      'conpcp': ('kg/m**2','Unknown'),
      'trdval': ('dval_m','Unkown'),
      'solflx': ('W/m**2','Solar Flux'),
      'cupcap': ('J/m**2','Unknown'),
      'irrflx': ('W/m**2','Unknown'),
      'slhflx': ('W/m**2','Unknown'),
      'sehflx': ('W/m**2','Unknown'),
      'totpcp': ('cm','Unknown'),
```

(continues on next page)

(continued from previous page)

```
      'bouflx': ('W/m**2','Unknown'),
      'totflx': ('W/m**2','Total Flux'),
      'irflux': ('W/m**2','Infrared Flux'),
      'liftcl': ('m','Lifting Condensation Level'),
      'ht_sfc': ('m/s','Surface Height'),
      'uustrs': ('N/m**2','Zonal Wind Stress'),
      'vvstrs': ('N/m**2','Meridional Wind Stress'),
      'wngust': ('m/s','Wind Gust'),
      'dwptdp': ('C','Dewpoint Depression'),
      'diverg': ('1/s','Divergence'),
      'absvor': ('1/s','Vorticity'),
      'womega': ('cPa/s','Vertical Velocity'),
      'stmfun': ('m**2/s','Stream Function'),
      'velpot': ('m**2/s','Velocity Potential'),
      'stacld': ('percent','Stable Cloud'),
      'concld': ('percent','Convective Cloud'),
      'clouds': ('percent','Total Cloud'),
    }

##########################################

   ##
   ##  input file specified on the command line
   ##  load the data into the numpy array
   ##

if len(sys.argv) == 2:

    # Store the input file and record number
    input_file = os.path.expandvars(sys.argv[1])
    tokens = os.path.basename(input_file).replace('-', '_').split('_');
    varname = tokens[0]
    nlons = int(tokens[4][4:7])  # Usually 360
    nlats = int(tokens[4][8:])   # Usually 181
    try:
        # Print some output to verify that this script ran
        print("Input File: " + repr(input_file))

        # Read input file
        data = np.float64(np.fromfile(input_file, '>f'))

        # Read and re-orient the data
        met_data = data[::-1].reshape(nlats, nlons)[:,::-1].copy()

        print("Data Shape: " + repr(met_data.shape))
```

(continues on next page)

(continued from previous page)

```
        print("Data Type:  " + repr(met_data.dtype))
        print("Data Range: " + repr(min(data)) + " to " + repr(max(data)) +
              " " + var_info[varname][0])
    except NameError:
        print("Trouble reading input file: " + input_file)
else:
    print("Must specify exactly one input file.")
    sys.exit(1)

########################################

   ##
   ##  create the metadata dictionary
   ##

for token in tokens:
   if(re.search("[0-9]{10,10}", token)):
       ymdh = dt.datetime.strptime(token[0:10],"%Y%m%d%H")
   elif(re.search("[0-9]{8,8}", token)):
       fhr = int(token) / 10000

init  = ymdh
valid = init + dt.timedelta(hours=fhr)
lead, rem = divmod((valid-init).total_seconds(), 3600)

attrs = {
   'valid': valid.strftime("%Y%m%d_%H%M%S"),
   'init':  init.strftime("%Y%m%d_%H%M%S"),
   'lead':  str(int(lead)),
   'accum': '00',

   'name':      varname,
   'long_name': var_info[varname][1],
   'level':     tokens[1]+'_'+tokens[2],
   'units':     var_info[varname][0],

   'grid': {
       'name': 'Global 1 Degree',
       'type' : 'LatLon',
       'lat_ll' :    -90.0,
       'lon_ll' :      0.0,
       'delta_lat' :   1.0,
       'delta_lon' :   1.0,
       'Nlat' :      nlats,
       'Nlon' :      nlons,
```

(continues on next page)

(continued from previous page)

```
   }
}

print("Attributes: " + repr(attrs))
```

Running METplus

This use case can be run two ways:

1) Passing in PointStat_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PointStat/PointStat_
↪python_embedding.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PointStat_python_embedding.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PointStat/PointStat_
↪python_embedding.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in point_stat_py_embed (relative to **OUTPUT_BASE**) and will contain the following files:

- point_stat_000000L_20200906_060000V.stat

Keywords

Note:

- PointStatToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-PointStat.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.22 PyEmbedIngest

7.1.32.22.1 PyEmbedIngest: Multiple Fields in One File

met_tool_wrapper/PyEmbedIngest/PyEmbedIngest_multi_field_one_file.conf

Scientific Objective

Converting file formats so data can be read by the MET tools. This use case demonstrates the ability to utilize two python embedding script calls to generate multiple fields in a single output file.

Datasets

Inputs: Canned ASCII data to test functionality

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 632) section for more information.

METplus Components

This use case utilizes the METplus PyEmbedIngest wrapper to generate a command to run the MET tool RegridDataPlane if all required files are found.

METplus Workflow

PyEmbedIngest is the only tool called in this example. It has one run time, but the time is not relevant because the files processed do not have any time information in the names.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PyEmbedIngest/PyEmbedIngest_multi_field_one_file.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PyEmbedIngest/
↪PyEmbedIngest_multi_field_one_file.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PyEmbedIngest


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2013022712
VALID_END = 2013022712
VALID_INCREMENT = 21600

LEAD_SEQ = 0


###
# PyEmbedIngest Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pyembedingest
###

PY_EMBED_INGEST_1_OUTPUT_DIR =
PY_EMBED_INGEST_1_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/PyEmbedIngest/test.nc

PY_EMBED_INGEST_1_SCRIPT_1 = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py
↪{INPUT_BASE}/met_test/data/python/fcst.txt FCST
PY_EMBED_INGEST_1_TYPE = NUMPY
PY_EMBED_INGEST_1_OUTPUT_GRID = G130


PY_EMBED_INGEST_2_OUTPUT_DIR =
PY_EMBED_INGEST_2_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/PyEmbedIngest/regrid_data_
↪plane.nc

PY_EMBED_INGEST_2_SCRIPT_1 = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py
↪{INPUT_BASE}/met_test/data/python/fcst.txt FCST
PY_EMBED_INGEST_2_OUTPUT_FIELD_NAME_1 = Forecast

PY_EMBED_INGEST_2_SCRIPT_2 = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py
↪{INPUT_BASE}/met_test/data/python/obs.txt OBS
PY_EMBED_INGEST_2_OUTPUT_FIELD_NAME_2 = Observation

PY_EMBED_INGEST_2_TYPE = NUMPY
PY_EMBED_INGEST_2_OUTPUT_GRID = G130
```

MET Configuration

None. RegridDataPlane does not use configuration files.

Python Embedding

This use case calls a Python script to read the input data. The Python script is stored in the MET repository: /path/to/MET/installation/share/met/python/read_ascii_numpy.py

read_ascii_numpy.py

Running METplus

This use case can be run two ways:

1) Passing in PyEmbedIngest_multi_field_one_file.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PyEmbedIngest/
↪PyEmbedIngest_multi_field_one_file.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PyEmbedIngest_multi_field_one_file.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PyEmbedIngest/
↪PyEmbedIngest_multi_field_one_file.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/PyEmbedIngest (relative to **OUTPUT_BASE**) and will contain the following file:

- test.nc
- regrid_data_plane.nc

Keywords

Note:

- PyEmbedIngestToolUseCase
- PythonEmbeddingFileUseCase
- RegridDataPlaneToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.22.2 PyEmbedIngest: Basic Use Case

met_tool_wrapper/PyEmbedIngest/PyEmbedIngest.conf

Scientific Objective

None. Simply converting file formats so data can be read by the MET tools.

Datasets

Inputs: Canned ASCII data to test functionality

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 636) section for more information.

METplus Components

This use case utilizes the METplus PyEmbedIngest wrapper to generate a command to run the MET tool RegridDataPlane if all required files are found.

METplus Workflow

PyEmbedIngest is the only tool called in this example. It has one run time, but the time is not relevant because the files processed do not have any time information in the names.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/PyEmbedIngest/PyEmbedIngest.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PyEmbedIngest/
↪PyEmbedIngest.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PyEmbedIngest


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2013022712
VALID_END = 2013022712
VALID_INCREMENT = 21600

LEAD_SEQ = 0


###
# PyEmbedIngest Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pyembedingest
###

PY_EMBED_INGEST_1_OUTPUT_DIR =
PY_EMBED_INGEST_1_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/PyEmbedIngest/test.nc

PY_EMBED_INGEST_1_SCRIPT = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py
↪{INPUT_BASE}/met_test/data/python/obs.txt OBS
PY_EMBED_INGEST_1_TYPE = NUMPY
PY_EMBED_INGEST_1_OUTPUT_GRID = G130


PY_EMBED_INGEST_2_OUTPUT_DIR =
PY_EMBED_INGEST_2_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/PyEmbedIngest/regrid_data_
↪plane.nc

PY_EMBED_INGEST_2_SCRIPT = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py
↪{INPUT_BASE}/met_test/data/python/fcst.txt FCST
PY_EMBED_INGEST_2_TYPE = NUMPY
PY_EMBED_INGEST_2_OUTPUT_GRID = G130
```

MET Configuration

None. RegridDataPlane does not use configuration files.

Python Embedding

This use case calls a Python script to read the input data. The Python script is stored in the MET repository: /path/to/MET/installation/share/met/python/read_ascii_numpy.py

read_ascii_numpy.py

Running METplus

This use case can be run two ways:

1) Passing in PyEmbedIngest.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PyEmbedIngest/
↪PyEmbedIngest.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PyEmbedIngest.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/PyEmbedIngest/
↪PyEmbedIngest.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/PyEmbedIngest (relative to **OUTPUT_BASE**) and will contain the following file:

- test.nc
- regrid_data_plane.nc

Keywords

Note:

- PyEmbedIngestToolUseCase
- PythonEmbeddingFileUseCase
- RegridDataPlaneToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.23 RegridDataPlane

7.1.32.23.1 RegridDataPlane: Process all fields

met_tool_wrapper/RegridDataPlane/RegridDataPlane_multi_field_one _file.conf

Scientific Objective

Simply regridding data to match a desired grid domain for comparisons. Process all fields in a single call to RegridDataPlane

Datasets

Forecast: WRF 3 hour precipitation accumulation and temperature

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 641) section for more information.

METplus Components

This use case utilizes the METplus RegridDataPlane wrapper to generate a command to run the MET tool RegridDataPlane if all required files are found.

METplus Workflow

RegridDataPlane is the only tool called in this example. It processes the following run time:

Init: 2005-08-07_0Z
Forecast lead: 3 hour

This use case regrids data to another domain specified with REGRID_DATA_PLANE_VERIF_GRID. This is done so that forecast and observation comparisons are done on the same grid. Many MET comparison tools have regridding capabilities built in. However, if the same file is read for comparisons multiple times, it is redundant to regrid that file each time. Running RegridDataPlane allows you to regrid once and use the output in many comparisons/evaluations.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/RegridDataPlane/RegridDataPlane_multi_field_one_file.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/RegridDataPlane/
```

(continues on next page)

(continued from previous page)

```
↪RegridDataPlane_multi_field_one_file.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = RegridDataPlane


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 1M

LEAD_SEQ = 3H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

OBS_REGRID_DATA_PLANE_RUN = True

OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst

OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/RegridDataPlane/multi_
```

(continues on next page)

(continued from previous page)

```
↪field_one_file

OBS_REGRID_DATA_PLANE_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%2H}.tm00_G212


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

REGRID_DATA_PLANE_ONCE_PER_FIELD = False

OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = APCP
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = L0
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = APCP_01

OBS_REGRID_DATA_PLANE_VAR2_INPUT_FIELD_NAME = TMP
OBS_REGRID_DATA_PLANE_VAR2_INPUT_LEVEL = L0


###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

MODEL = QPF
OBTYPE = QPE

REGRID_DATA_PLANE_VERIF_GRID={INPUT_BASE}/met_test/data/sample_obs/ST2ml/ST2ml2005080703.Grb_
↪G212

REGRID_DATA_PLANE_METHOD = BUDGET

REGRID_DATA_PLANE_WIDTH = 2

REGRID_DATA_PLANE_GAUSSIAN_DX =

REGRID_DATA_PLANE_GAUSSIAN_RADIUS =
```

MET Configuration

None. RegridDataPlane does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in RegridDataPlane_multi_field_one_file.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/RegridDataPlane/
↪RegridDataPlane_multi_field_one_file.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in RegridDataPlane_multi_field_one_file.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/RegridDataPlane/
↪RegridDataPlane_multi_field_one_file.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/RegridDataPlane (relative to **OUTPUT_BASE**) and will contain the following files:

- multi_field_one_file/2005080700/wrfprs_ruc13_03.tm00_G212

Keywords

Note:

- RegridDataPlaneToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-RegridDataPlane.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.23.2 RegridDataPlane: Basic Use Case

met_tool_wrapper/RegridDataPlane/RegridDataPlane.conf

Scientific Objective

Simply regridding data to match a desired grid domain for comparisons.

Datasets

Observations: Stage 2 NetCDF 1-hour Precipitation Accumulation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 645) section for more information.

METplus Components

This use case utilizes the METplus RegridDataPlane wrapper to generate a command to run the MET tool RegridDataPlane if all required files are found.

METplus Workflow

RegridDataPlane is the only tool called in this example. It processes the following run time:

Init: 2005-08-07_0Z
Forecast lead: 3 hour

This use case regrids data to another domain specified with REGRID_DATA_PLANE_VERIF_GRID. This is done so that forecast and observation comparisons are done on the same grid. Many MET comparison tools have regridding capabilities built in. However, if the same file is read for comparisons multiple times, it is redundant to regrid that file each time. Running RegridDataPlane allows you to regrid once and use the output in many comparisons/evaluations.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/RegridDataPlane/RegridDataPlane.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/RegridDataPlane/
↪RegridDataPlane.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = RegridDataPlane


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
```

(continues on next page)

(continued from previous page)

```
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 1M

LEAD_SEQ = 3H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

OBS_REGRID_DATA_PLANE_RUN = True

OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_obs/ST2ml

OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/regrid_data_plane

OBS_REGRID_DATA_PLANE_TEMPLATE = ST2ml{valid?fmt=%Y%m%d%H}.Grb_G212


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = APCP
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = A01
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = APCP_01

REGRID_DATA_PLANE_ONCE_PER_FIELD = True


###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
```

(continues on next page)

(continued from previous page)

```
###

MODEL = QPF
OBTYPE = QPE

REGRID_DATA_PLANE_VERIF_GRID={INPUT_BASE}/met_test/data/sample_fcst/2005080700/wrfprs_ruc13_
↪12.tm00_G212

REGRID_DATA_PLANE_METHOD = BUDGET

REGRID_DATA_PLANE_WIDTH = 2

REGRID_DATA_PLANE_GAUSSIAN_DX =

REGRID_DATA_PLANE_GAUSSIAN_RADIUS =
```

MET Configuration

None. RegridDataPlane does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in RegridDataPlane.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/RegridDataPlane/
↪RegridDataPlane.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in RegridDataPlane.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/RegridDataPlane/
↪RegridDataPlane.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in regrid_data_plane (relative to **OUTPUT_BASE**) and will contain the following file:

- ST2ml2005080703.Grb_G212

Keywords

Note:

- RegridDataPlaneToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-RegridDataPlane.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.23.3 RegridDataPlane: Run once per field

met_tool_wrapper/RegridDataPlane/RegridDataPlane_multi_field _multi_file.conf

Scientific Objective

Simply regridding data to match a desired grid domain for comparisons. Process each field separately and write a file for each.

Datasets

Forecast: WRF 3 hour precipitation accumulation and temperature

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 650) section for more information.

METplus Components

This use case utilizes the METplus RegridDataPlane wrapper to generate a command to run the MET tool RegridDataPlane if all required files are found.

METplus Workflow

RegridDataPlane is the only tool called in this example. It processes the following run time:

Init: 2005-08-07_0Z
Forecast lead: 3 hour

This use case regrids data to another domain specified with REGRID_DATA_PLANE_VERIF_GRID. This is done so that forecast and observation comparisons are done on the same grid. Many MET comparison tools have regridding capabilities built in. However, if the same file is read for comparisons multiple times, it is redundant to regrid that file each time. Running RegridDataPlane allows you to regrid once and use the output in many comparisons/evaluations.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/RegridDataPlane/RegridDataPlane_multi_field_multi_file.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/RegridDataPlane/
```

(continues on next page)

(continued from previous page)

```
↪RegridDataPlane_multi_field_multi_file.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = RegridDataPlane


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 1M

LEAD_SEQ = 3H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

OBS_REGRID_DATA_PLANE_RUN = True

OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%2H}.tm00_
↪G212
```

(continues on next page)

(continued from previous page)

```
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/RegridDataPlane/multi_
↪field_multi_file
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_{obs_name}_{lead?fmt=%2H}.
↪tm00_G212


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

REGRID_DATA_PLANE_ONCE_PER_FIELD = True

OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = APCP
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = L0
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = APCP_01

OBS_REGRID_DATA_PLANE_VAR2_INPUT_FIELD_NAME = TMP
OBS_REGRID_DATA_PLANE_VAR2_INPUT_LEVEL = L0


###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

MODEL = QPF
OBTYPE = QPE

REGRID_DATA_PLANE_VERIF_GRID={INPUT_BASE}/met_test/data/sample_obs/ST2ml/ST2ml2005080703.Grb_
↪G212

REGRID_DATA_PLANE_METHOD = BUDGET

REGRID_DATA_PLANE_WIDTH = 2

REGRID_DATA_PLANE_GAUSSIAN_DX =

REGRID_DATA_PLANE_GAUSSIAN_RADIUS =
```

MET Configuration

None. RegridDataPlane does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in RegridDataPlane_multi_field_multi_file.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/RegridDataPlane/
↪RegridDataPlane_multi_field_multi_file.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in RegridDataPlane_multi_field_multi_file.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/RegridDataPlane/
↪RegridDataPlane_multi_field_multi_file.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/RegridDataPlane (relative to **OUTPUT_BASE**) and will contain the following files:

- multi_field_multi_file/2005080700/wrfprs_APCP_03.tm00_G212
- multi_field_multi_file/2005080700/wrfprs_TMP_03.tm00_G212

Keywords

Note:

- RegridDataPlaneToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-RegridDataPlane.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.23.4 RegridDataPlane: Using Python Embedding

met_tool_wrapper/RegridDataPlane/RegridDataPlane_python _embedding.conf

Scientific Objective

None. Simply regridding data to match a desired grid domain for comparisons.

Datasets

Forecast: ASCII sample file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 654) section for more information.

METplus Components

This use case utilizes the METplus RegridDataPlane wrapper to generate a command to run the MET tool RegridDataPlane if all required files are found.

METplus Workflow

RegridDataPlane is the only tool called in this example. It processes a single run time, but the data does not contain any time information in the filename, so the run time is irrelevant.

This use case regrids data to another domain specified with REGRID_DATA_PLANE_VERIF_GRID. This is done so that forecast and observation comparisons are done on the same grid. Many MET comparison tools have regridding capabilities built in. However, if the same file is read for comparisons multiple times, it is redundant to regrid that file each time. Running RegridDataPlane allows you to regrid once and use the output in many comparisons/evaluations.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/RegridDataPlane/RegridDataPlane_python_embedding.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/RegridDataPlane/
↪RegridDataPlane_python_embedding.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = RegridDataPlane


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
```

(continues on next page)

(continued from previous page)

```
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT=43200

LEAD_SEQ = 3


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

OBS_REGRID_DATA_PLANE_RUN = True

OBS_REGRID_DATA_PLANE_INPUT_DIR =
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/RegridDataPlane/regrid_py
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = numpy_data.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

REGRID_DATA_PLANE_ONCE_PER_FIELD = True

OBS_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py {INPUT_BASE}/
↪met_test/data/python/obs.txt OBS
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = OBS


###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
```

(continues on next page)

(continued from previous page)

```
###

MODEL = FCST
OBTYPE = OBS

REGRID_DATA_PLANE_VERIF_GRID = G130

REGRID_DATA_PLANE_METHOD = BUDGET

REGRID_DATA_PLANE_WIDTH = 2
```

MET Configuration

None. RegridDataPlane does not use configuration files.

Python Embedding

This use case calls a Python script to read the input data. The Python script is stored in the MET repository: /path/to/MET/installation/share/met/python/read_ascii_numpy.py

read_ascii_numpy.py

Running METplus

This use case can be run two ways:

1) Passing in RegridDataPlane_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/RegridDataPlane/
↪RegridDataPlane_python_embedding.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in RegridDataPlane_python_embedding.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/RegridDataPlane/
↪RegridDataPlane_python_embedding.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/RegridDataPlane/regrid_py (relative to **OUTPUT_BASE**) and will contain the following file:

- numpy_data.nc

Keywords

Note:

- RegridDataPlaneToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-RegridDataPlane.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.24 SeriesAnalysis

7.1.32.24.1 SeriesAnalysis: Basic Use Case

met_tool_wrapper/SeriesAnalysis/SeriesAnalysis.conf

Scientific Objective

Compare forecasts for 3-hour precipitation accumulations to observed 3-hour accumulation. These comparisons are made through generating statistics of the results.

Datasets

Forecast: WRF 3 hour precipitation accumulation
Observation: MU 3 hour precipitation accumulation

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 664) section for more information.

METplus Components

This use case utilizes the METplus SeriesAnalysis wrapper to search for files that are valid at a given run time and generates a command to run the MET tool series_analysis if all required files are found.

METplus Workflow

SeriesAnalysis is the only tool called in this example. It processes the following run times:

Init: 2005-08-07_0Z
Forecast lead: 12 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/SeriesAnalysis/SeriesAnalysis.conf

```
[config]

# Documentation for this use case can be found at
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/SeriesAnalysis/
↪SeriesAnalysis.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = SeriesAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 12H

LEAD_SEQ = 12, 9, 6


SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False

SERIES_ANALYSIS_CUSTOM_LOOP_LIST =

###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
```

(continues on next page)

(continued from previous page)

```
###

FCST_SERIES_ANALYSIS_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%2H}.tm00_
↪G212

OBS_SERIES_ANALYSIS_INPUT_DIR = {INPUT_BASE}/met_test/new
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = ST2ml{valid?fmt=%Y%m%d%H}_A03h.nc

SERIES_ANALYSIS_TC_STAT_INPUT_DIR =
SERIES_ANALYSIS_TC_STAT_INPUT_TEMPLATE =

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/SeriesAnalysis
SERIES_ANALYSIS_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}_sa.nc

SERIES_ANALYSIS_CLIMO_MEAN_INPUT_DIR =
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_TEMPLATE =

SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR =
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_TEMPLATE =


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

#FCST_CAT_THRESH =
#OBS_CAT_THRESH =

FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A03

OBS_VAR1_NAME = APCP_03
OBS_VAR1_LEVELS = "(*,*)"

BOTH_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2


###
# SeriesAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#seriesanalysis
###

MODEL = WRF
```

(continues on next page)

(continued from previous page)

```
OBTYPE = MC_PCP

#LOG_SERIES_ANALYSIS_VERBOSITY = 2

SERIES_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/SeriesAnalysisConfig_wrapped

SERIES_ANALYSIS_IS_PAIRED = False

#SERIES_ANALYSIS_DESC =

#SERIES_ANALYSIS_CAT_THRESH =

#SERIES_ANALYSIS_VLD_THRESH =

#SERIES_ANALYSIS_BLOCK_SIZE =

#SERIES_ANALYSIS_REGRID_TO_GRID =
#SERIES_ANALYSIS_REGRID_METHOD =
#SERIES_ANALYSIS_REGRID_WIDTH =
#SERIES_ANALYSIS_REGRID_VLD_THRESH =
#SERIES_ANALYSIS_REGRID_SHAPE =
#SERIES_ANALYSIS_REGRID_CONVERT =
#SERIES_ANALYSIS_REGRID_CENSOR_THRESH =
#SERIES_ANALYSIS_REGRID_CENSOR_VAL =

#SERIES_ANALYSIS_CLIMO_MEAN_FILE_NAME =
#SERIES_ANALYSIS_CLIMO_MEAN_FIELD =
#SERIES_ANALYSIS_CLIMO_MEAN_REGRID_METHOD =
#SERIES_ANALYSIS_CLIMO_MEAN_REGRID_WIDTH =
#SERIES_ANALYSIS_CLIMO_MEAN_REGRID_VLD_THRESH =
#SERIES_ANALYSIS_CLIMO_MEAN_REGRID_SHAPE =
#SERIES_ANALYSIS_CLIMO_MEAN_TIME_INTERP_METHOD =
#SERIES_ANALYSIS_CLIMO_MEAN_MATCH_MONTH =
#SERIES_ANALYSIS_CLIMO_MEAN_DAY_INTERVAL =
#SERIES_ANALYSIS_CLIMO_MEAN_HOUR_INTERVAL =
#SERIES_ANALYSIS_CLIMO_MEAN_FILE_TYPE =

#SERIES_ANALYSIS_CLIMO_STDEV_FILE_NAME =
#SERIES_ANALYSIS_CLIMO_STDEV_FIELD =
#SERIES_ANALYSIS_CLIMO_STDEV_REGRID_METHOD =
#SERIES_ANALYSIS_CLIMO_STDEV_REGRID_WIDTH =
#SERIES_ANALYSIS_CLIMO_STDEV_REGRID_VLD_THRESH =
#SERIES_ANALYSIS_CLIMO_STDEV_REGRID_SHAPE =
#SERIES_ANALYSIS_CLIMO_STDEV_TIME_INTERP_METHOD =
#SERIES_ANALYSIS_CLIMO_STDEV_MATCH_MONTH =
```

(continues on next page)

(continued from previous page)

```
#SERIES_ANALYSIS_CLIMO_STDEV_DAY_INTERVAL =
#SERIES_ANALYSIS_CLIMO_STDEV_HOUR_INTERVAL =
#SERIES_ANALYSIS_CLIMO_STDEV_FILE_TYPE =

#SERIES_ANALYSIS_CLIMO_CDF_BINS =
#SERIES_ANALYSIS_CLIMO_CDF_CENTER_BINS =
#SERIES_ANALYSIS_CLIMO_CDF_DIRECT_PROB =

#SERIES_ANALYSIS_HSS_EC_VALUE =

#FCST_SERIES_ANALYSIS_PROB_THRESH =

#SERIES_ANALYSIS_OUTPUT_STATS_FHO =
#SERIES_ANALYSIS_OUTPUT_STATS_CTC =
#SERIES_ANALYSIS_OUTPUT_STATS_CTS =
#SERIES_ANALYSIS_OUTPUT_STATS_MCTC =
#SERIES_ANALYSIS_OUTPUT_STATS_MCTS =

SERIES_ANALYSIS_OUTPUT_STATS_CNT = TOTAL, RMSE, FBAR, OBAR

#SERIES_ANALYSIS_OUTPUT_STATS_SL1L2 =
#SERIES_ANALYSIS_OUTPUT_STATS_SAL1L2 =
#SERIES_ANALYSIS_OUTPUT_STATS_PCT =
#SERIES_ANALYSIS_OUTPUT_STATS_PSTD =
#SERIES_ANALYSIS_OUTPUT_STATS_PJC =
#SERIES_ANALYSIS_OUTPUT_STATS_PRC =

#SERIES_ANALYSIS_MASK_GRID =
#SERIES_ANALYSIS_MASK_POLY =


###
# SeriesAnalysis Plotting
###

SERIES_ANALYSIS_GENERATE_PLOTS = no

PLOT_DATA_PLANE_TITLE =

SERIES_ANALYSIS_GENERATE_ANIMATIONS = no
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *SeriesAnalysis MET Configuration* (page 232) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Series-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
//
//desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
//obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
```

(continues on next page)

(continued from previous page)

```
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

censor_thresh = [];
censor_val    = [];
//cat_thresh =
${METPLUS_CAT_THRESH}
cnt_thresh    = [ NA ];
cnt_logic     = UNION;

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_CAT_THRESH}
   ${METPLUS_FCST_FIELD}
}
obs = {
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
```

(continues on next page)

(continued from previous page)

```
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

//
// Number of grid points to be processed concurrently.  Set smaller to use
// less memory but increase the number of passes through the data.
//
//block_size =
${METPLUS_BLOCK_SIZE}

//
// Ratio of valid matched pairs to compute statistics for a grid point
//
//vld_thresh =
${METPLUS_VLD_THRESH}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_stats = {
${METPLUS_OUTPUT_STATS_DICT}

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;
```

(continues on next page)

(continued from previous page)

```
tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in SeriesAnalysis.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/SeriesAnalysis/
↪SeriesAnalysis.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in SeriesAnalysis.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/SeriesAnalysis/
↪SeriesAnalysis.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/SeriesAnalysis (relative to **OUTPUT_BASE**) and will contain the following file:

- 2005080700_sa.nc

Keywords

Note:

- SeriesAnalysisUseCase
- DiagnosticsUseCase
- ListExpansionFeatureUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-SeriesAnalysis.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.24.2 SeriesAnalysis: Using Python Embedding

met_tool_wrapper/SeriesAnalysis/SeriesAnalysis_python _embedding.conf

Scientific Objective

None. This is a demonstration of using python embedding to pass and read in external files, which have a data format that MET would not otherwise be able to parse.

Datasets

Forecast: Dummy text files found in the MET shared directory
Observation: Dummy text files found in the MET shared directory

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 672) section for more information.

METplus Components

This use case utilizes the METplus SeriesAnalysis wrapper to search for files as determined by the Python script.

METplus Workflow

SeriesAnalysis is the only tool called in this example. It processes simple text files with no timining or meteorological information to demonstrate how SeriesAnalysis can be run utilizing Python Embedding.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/SeriesAnalysis/SeriesAnalysis_python_embedding.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/SeriesAnalysis/
↪SeriesAnalysis_python_embedding.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###
```

(continues on next page)

(continued from previous page)

```
PROCESS_LIST = SeriesAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 12H

LEAD_SEQ = 12

SERIES_ANALYSIS_CUSTOM_LOOP_LIST =


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_SERIES_ANALYSIS_INPUT_DIR = {INPUT_BASE}/met_test/data/python
FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = fcst.txt, fcst.txt
FCST_SERIES_ANALYSIS_INPUT_DATATYPE = PYTHON_NUMPY

OBS_SERIES_ANALYSIS_INPUT_DIR = {INPUT_BASE}/met_test/data/python
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = obs.txt, obs.txt
OBS_SERIES_ANALYSIS_INPUT_DATATYPE = PYTHON_NUMPY

SERIES_ANALYSIS_CLIMO_MEAN_INPUT_DIR =
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_TEMPLATE =
```

(continues on next page)

(continued from previous page)

```
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR =
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_TEMPLATE =

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/SeriesAnalysis
SERIES_ANALYSIS_OUTPUT_TEMPLATE = python_sa.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

FCST_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py MET_PYTHON_
↪INPUT_ARG FCST

OBS_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/examples/read_ascii_numpy.py MET_PYTHON_
↪INPUT_ARG OBS


###
# SeriesAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#seriesanalysis
###

SERIES_ANALYSIS_DESC =

SERIES_ANALYSIS_CAT_THRESH =

SERIES_ANALYSIS_VLD_THRESH =

SERIES_ANALYSIS_BLOCK_SIZE =

SERIES_ANALYSIS_CTS_LIST =

SERIES_ANALYSIS_REGRID_TO_GRID =
SERIES_ANALYSIS_REGRID_METHOD =
SERIES_ANALYSIS_REGRID_WIDTH =
SERIES_ANALYSIS_REGRID_VLD_THRESH =
SERIES_ANALYSIS_REGRID_SHAPE =

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False

#LOG_SERIES_ANALYSIS_VERBOSITY = 2

SERIES_ANALYSIS_IS_PAIRED = False
```

(continues on next page)

(continued from previous page)

```
SERIES_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/SeriesAnalysisConfig_wrapped

SERIES_ANALYSIS_STAT_LIST = TOTAL, RMSE, FBAR, OBAR

MODEL = PYTHON

OBTYPE = ANALYS
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *SeriesAnalysis MET Configuration* (page 232) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Series-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
//
//desc =
${METPLUS_DESC}

//
```

(continues on next page)

(continued from previous page)

```
// Output observation type to be written
//
//obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

censor_thresh = [];
censor_val    = [];
//cat_thresh =
${METPLUS_CAT_THRESH}
cnt_thresh    = [ NA ];
cnt_logic     = UNION;

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_CAT_THRESH}
   ${METPLUS_FCST_FIELD}
}
obs = {
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}
```

(continues on next page)

(continued from previous page)

```
//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

//
// Number of grid points to be processed concurrently.  Set smaller to use
// less memory but increase the number of passes through the data.
//
//block_size =
${METPLUS_BLOCK_SIZE}

//
// Ratio of valid matched pairs to compute statistics for a grid point
//
//vld_thresh =
${METPLUS_VLD_THRESH}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Statistical output types
//
//output_stats = {
${METPLUS_OUTPUT_STATS_DICT}

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case calls a Python script to read the input data. The Python script is stored in the MET repository: /path/to/MET/installation/share/met/python/read_ascii_numpy.py

read_ascii_numpy.py

Running METplus

This use case can be run two ways:

1) Passing in SeriesAnalysis_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/SeriesAnalysis/
↪SeriesAnalysis_python_embedding.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in SeriesAnalysis_python_embedding.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/SeriesAnalysis/
↪SeriesAnalysis_python_embedding.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/SeriesAnalysis (relative to **OUTPUT_BASE**) and will contain the following file:

- python_sa.nc

Keywords

Note:

- SeriesAnalysisUseCase
- PythonEmbeddingFileUseCase
- DiagnosticsUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-SeriesAnalysis.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.25 StatAnalysis

7.1.32.25.1 StatAnalysis: Using Python Embedding

met_tool_wrapper/StatAnalysis/StatAnalysis_python_embedding.conf

Scientific Objective

This demonstrates how the Stat-Analysis tool can tie together results from other MET tools (including Point-Stat, GridStat, EnsembleStat, and WaveletStat) and provide summary statistical information. Matched pair records are passed into Stat-Analysis using python embedding.

Datasets

WRF ARW point_stat output STAT files:

. . . met_test/new

point_stat_120000L_20050807_120000V.stat

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 680) section for more information.

Data Source: WRF

METplus Components

This use case utilizes the METplus StatAnalysis wrapper to search for files that are valid at a given run time and generate a command to run the MET tool stat_analysis.

METplus Workflow

StatAnalysis is the only tool called in this example. It processes the following run times:

Valid: 2005-08-07_00Z

Forecast lead: 12 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/StatAnalysis/StatAnalysis.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/StatAnalysis/
↪StatAnalysis_python_embedding.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = StatAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG=20070331
VALID_END=20070331
VALID_INCREMENT = 12H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###
```

(continues on next page)

(continued from previous page)

```
MODEL1 = WRF
MODEL1_OBTYPE = ADPUPA
MODEL1_STAT_ANALYSIS_LOOKIN_DIR = python {MET_INSTALL_DIR}/share/met/python/examples/read_
↪ascii_mpr.py {INPUT_BASE}/met_test/out/point_stat/point_stat_360000L_20070331_120000V.stat

MODEL1_STAT_ANALYSIS_OUT_STAT_TEMPLATE = {model?fmt=%s}_{obtype?fmt=%s}_valid{valid?fmt=%Y%m
↪%d}_fcstvalidhour{valid_hour?fmt=%H}0000Z_out_stat.stat

STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/StatAnalysis_python_embedding


###
# StatAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#statanalysis
###

#LOG_STAT_ANALYSIS_VERBOSITY = 2

STAT_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/STATAnalysisConfig_wrapped

STAT_ANALYSIS_JOB_NAME = aggregate_stat
STAT_ANALYSIS_JOB_ARGS = -out_line_type sl1l2 -by FCST_VAR -out_stat [out_stat_file]

MODEL_LIST = {MODEL1}
DESC_LIST =
FCST_LEAD_LIST =
OBS_LEAD_LIST =
FCST_VALID_HOUR_LIST = 12
FCST_INIT_HOUR_LIST =
OBS_VALID_HOUR_LIST =
OBS_INIT_HOUR_LIST =
FCST_VAR_LIST =
OBS_VAR_LIST =
FCST_UNITS_LIST =
OBS_UNITS_LIST =
FCST_LEVEL_LIST =
OBS_LEVEL_LIST =
VX_MASK_LIST =
INTERP_MTHD_LIST =
INTERP_PNTS_LIST =
FCST_THRESH_LIST =
OBS_THRESH_LIST =
COV_THRESH_LIST =
ALPHA_LIST =
```

(continues on next page)

(continued from previous page)

```
LINE_TYPE_LIST = MPR

GROUP_LIST_ITEMS = FCST_INIT_HOUR_LIST
LOOP_LIST_ITEMS = FCST_VALID_HOUR_LIST, MODEL_LIST
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *StatAnalysis MET Configuration* (page 246) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// STAT-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Filtering input STAT lines by the contents of each column
//
${METPLUS_MODEL}
${METPLUS_DESC}

${METPLUS_FCST_LEAD}
${METPLUS_OBS_LEAD}

${METPLUS_FCST_VALID_BEG}
${METPLUS_FCST_VALID_END}
${METPLUS_FCST_VALID_HOUR}

${METPLUS_OBS_VALID_BEG}
${METPLUS_OBS_VALID_END}
${METPLUS_OBS_VALID_HOUR}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_FCST_INIT_BEG}
${METPLUS_FCST_INIT_END}
${METPLUS_FCST_INIT_HOUR}

${METPLUS_OBS_INIT_BEG}
${METPLUS_OBS_INIT_END}
${METPLUS_OBS_INIT_HOUR}

${METPLUS_FCST_VAR}
${METPLUS_OBS_VAR}

${METPLUS_FCST_UNITS}
${METPLUS_OBS_UNITS}

${METPLUS_FCST_LEVEL}
${METPLUS_OBS_LEVEL}

${METPLUS_OBTYPE}

${METPLUS_VX_MASK}

${METPLUS_INTERP_MTHD}

${METPLUS_INTERP_PNTS}

${METPLUS_FCST_THRESH}
${METPLUS_OBS_THRESH}
${METPLUS_COV_THRESH}

${METPLUS_ALPHA}

${METPLUS_LINE_TYPE}

column = [];

weight = [];

////////////////////////////////////////////////////////////////////////////////

//
// Array of STAT-Analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Confidence interval settings
//
out_alpha = 0.05;

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// WMO mean computation logic
//
wmo_sqrt_stats   = [ "CNT:FSTDEV",  "CNT:OSTDEV",  "CNT:ESTDEV",
                     "CNT:RMSE",    "CNT:RMSFA",   "CNT:RMSOA",
                     "VCNT:FS_RMS", "VCNT:OS_RMS", "VCNT:RMSVE",
                     "VCNT:FSTDEV", "VCNT:OSTDEV" ];

wmo_fisher_stats = [ "CNT:PR_CORR", "CNT:SP_CORR",
                     "CNT:KT_CORR", "CNT:ANOM_CORR" ];

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;
vif_flag       = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case calls a Python script to read matched pair lines from an input source. The Python script is stored in the MET repository: /path/to/MET/installation/share/met/python/read_ascii_mpr.py

read_ascii_mpr.py

Running METplus

It is recommended to run this use case by:

Passing in StatAnalysis_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/StatAnalysis_python_embedding.conf -c /path/to/user_system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/StatAnalysis_python_embedding (relative to **OUTPUT_BASE**) and will contain the following file:

- WRF_ADPSFC_valid20050807_fcstvalidhour120000Z_out_stat.stat

Keywords

Note:

- StatAnalysisToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-StatAnalysis.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.25.2 StatAnalysis: Basic Use Case

met_tool_wrapper/StatAnalysis/StatAnalysis.conf

Scientific Objective

This demonstrates how the Stat-Analysis tool can tie together results from other MET tools (including Point-Stat, GridStat, EnsembleStat, and WaveletStat) and provide summary statistical information.

Datasets

WRF ARW grid_stat output STAT files:

. . . met_test/out/grid_stat/

grid_stat_120000L_20050807_120000V.stat
grid_stat_APCP_12_120000L_20050807_120000V.stat
grid_stat_APCP_24_240000L_20050808_000000V.stat
grid_stat_POP_12_1080000L_20050808_000000V.stat

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 687) section for more information.
Data Source: WRF

METplus Components

This use case utilizes the METplus StatAnalysis wrapper to search for files that are valid at a given run time and generate a command to run the MET tool stat_analysis.

METplus Workflow

StatAnalysis is the only tool called in this example. It processes the following run times:

Valid: 2005-08-07_00Z
Forecast lead: 12 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/StatAnalysis/StatAnalysis.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/StatAnalysis/
↪StatAnalysis.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = StatAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG=2005080700
VALID_END=2005080700
VALID_INCREMENT = 12H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

MODEL1 = WRF
MODEL1_OBTYPE = ANALYS
MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {INPUT_BASE}/met_test/out/grid_stat
MODEL1_STAT_ANALYSIS_DUMP_ROW_TEMPLATE = {fcst_valid_hour?fmt=%H}Z/{MODEL1}/{MODEL1}_{valid?
↪fmt=%Y%m%d%H}.stat

STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/stat_analysis
STAT_ANALYSIS_OUTPUT_TEMPLATE = job.out


###
# StatAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#statanalysis
###

#LOG_STAT_ANALYSIS_VERBOSITY = 2

STAT_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/STATAnalysisConfig_wrapped

#STAT_ANALYSIS_HSS_EC_VALUE =

STAT_ANALYSIS_JOB1 = -job filter -dump_row [dump_row_file]

MODEL_LIST = {MODEL1}
DESC_LIST =
FCST_LEAD_LIST =
OBS_LEAD_LIST =
```

(continues on next page)

(continued from previous page)

```
FCST_VALID_HOUR_LIST = 12
FCST_INIT_HOUR_LIST = 00, 12
OBS_VALID_HOUR_LIST =
OBS_INIT_HOUR_LIST =
FCST_VAR_LIST = TMP
OBS_VAR_LIST =
FCST_UNITS_LIST =
OBS_UNITS_LIST =
FCST_LEVEL_LIST =
OBS_LEVEL_LIST =
VX_MASK_LIST =
INTERP_MTHD_LIST =
INTERP_PNTS_LIST =
FCST_THRESH_LIST =
OBS_THRESH_LIST =
COV_THRESH_LIST =
ALPHA_LIST =
LINE_TYPE_LIST =

GROUP_LIST_ITEMS = FCST_INIT_HOUR_LIST
LOOP_LIST_ITEMS = FCST_VALID_HOUR_LIST, MODEL_LIST
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *StatAnalysis MET Configuration* (page 246) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// STAT-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Filtering input STAT lines by the contents of each column
//
${METPLUS_MODEL}
${METPLUS_DESC}

${METPLUS_FCST_LEAD}
${METPLUS_OBS_LEAD}

${METPLUS_FCST_VALID_BEG}
${METPLUS_FCST_VALID_END}
${METPLUS_FCST_VALID_HOUR}

${METPLUS_OBS_VALID_BEG}
${METPLUS_OBS_VALID_END}
${METPLUS_OBS_VALID_HOUR}

${METPLUS_FCST_INIT_BEG}
${METPLUS_FCST_INIT_END}
${METPLUS_FCST_INIT_HOUR}

${METPLUS_OBS_INIT_BEG}
${METPLUS_OBS_INIT_END}
${METPLUS_OBS_INIT_HOUR}

${METPLUS_FCST_VAR}
${METPLUS_OBS_VAR}

${METPLUS_FCST_UNITS}
${METPLUS_OBS_UNITS}

${METPLUS_FCST_LEVEL}
${METPLUS_OBS_LEVEL}

${METPLUS_OBTYPE}

${METPLUS_VX_MASK}

${METPLUS_INTERP_MTHD}

${METPLUS_INTERP_PNTS}

${METPLUS_FCST_THRESH}
${METPLUS_OBS_THRESH}
${METPLUS_COV_THRESH}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_ALPHA}

${METPLUS_LINE_TYPE}

column = [];

weight = [];

////////////////////////////////////////////////////////////////////////////////

//
// Array of STAT-Analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
out_alpha = 0.05;

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// WMO mean computation logic
//
wmo_sqrt_stats   = [ "CNT:FSTDEV",  "CNT:OSTDEV",  "CNT:ESTDEV",
                     "CNT:RMSE",    "CNT:RMSFA",   "CNT:RMSOA",
                     "VCNT:FS_RMS", "VCNT:OS_RMS", "VCNT:RMSVE",
                     "VCNT:FSTDEV", "VCNT:OSTDEV" ];

wmo_fisher_stats = [ "CNT:PR_CORR", "CNT:SP_CORR",
                     "CNT:KT_CORR", "CNT:ANOM_CORR" ];

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;
vif_flag       = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

It is recommended to run this use case by:

Passing in StatAnalysis.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/StatAnalysis.conf -c /path/to/user_system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in stat_analysis/12Z/WRF (relative to **OUTPUT_BASE**) and will contain the following file:

- WRF_2005080712.stat

Keywords

Note:

- StatAnalysisToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-StatAnalysis.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.26 TCDiag

7.1.32.26.1 TCDiag: Basic Use Case

met_tool_wrapper/TCDiag/TCDiag.conf

Overview

This use case illustrates the use of tc_diag tool, which is currently considered a beta-level release that lacks full functionality. The use case illustrates running the tc_diag tool for a tropical cyclone forecast case and generating intermediate NetCDF output files of the input model's data transformed onto an azimuth-range grid. When the full functionality of the tc_diag tool is released in MET v12.0.0, this use case will also output environmental diagnostics computed from callable Python scripts.

The diagnostics are computed on a range-azimuth grid that follows the projected storm track. For inputs, it uses 0.25 deg gridded GRIB files from the a retrospective reforecast of the Global Forecast System (GFS). For the track, it uses the GFS's predicted track to ensure that the model's simulated storm doesn't contaminate the diagnostics result as a result of the model's simulated storm being mistaken for environmental factors. (Note: a future version of the tc_diag tool will include removal of the model's vortex, allowing diagnostics to be computed along any arbitrarily defined track.)

Novel aspects of this use case:

- This is the first example use case to run the tc_diag tool.
- Example of running for a single tropical cyclone forecast case from Tropical Storm Bret (2023) using GFS data.

Scientific Objective

Generate intermediate data files, in which the input model's data have been transformed to a range-azimuth grid, in preparation for further diagnostic calculations using Python-based routines.

Datasets

Forecast: GFS grib files

Track: a-deck file (Automated Tropical Cyclone Forecast System format)

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 697) section for more information.

Data source: Users may obtain real-time data from the deterministic GFS runs from NOAA's NOMADS server: https://nomads.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/gfs.YYYYMMDD/ZZ/atmos/ where YYYYMMDD is the date (4-digit year, 2-digit month, 2-digit day), ZZ is the initialization hour of the desired model cycle (00, 06, 12, 18).

METplus Components

This use case utilizes the METplus TCDiag wrapper to search for the desired ADECK file and forecast files that correspond to the track. It generates a command to run tc_diag if all required files are found.

METplus Workflow

TCDiag is the only tool called in this example. It processes the following run times:

Init: 2023-06-20 0000Z **Forecast lead:** 0, 6, and 12 hours

METplus Configuration

parm/use_cases/met_tool_wrapper/TCDiag/TCDiag.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/TCDiag/TCDiag.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCDiag


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2023062012
INIT_END = 2023062012
INIT_INCREMENT = 21600
LEAD_SEQ = 0, 6, 12

###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

TC_DIAG_DECK_INPUT_DIR = {INPUT_BASE}/met_test/new/tc_data/adeck
TC_DIAG_DECK_TEMPLATE = subset.aal03{date?fmt=%Y}.dat
```

(continues on next page)

(continued from previous page)

```
TC_DIAG_INPUT1_DIR = {INPUT_BASE}/met_test/new/model_data/grib2/gfs
TC_DIAG_INPUT1_TEMPLATE = subset.gfs.t12z.pgrb2.0p50.f*
TC_DIAG_INPUT1_DOMAIN = parent
TC_DIAG_INPUT1_TECH_ID_LIST = AVNO


TC_DIAG_INPUT2_DIR = {INPUT_BASE}/met_test/new/model_data/grib2/gfs
TC_DIAG_INPUT2_TEMPLATE = subset.gfs.t12z.pgrb2.0p50.f*
TC_DIAG_INPUT2_DOMAIN = nest
TC_DIAG_INPUT2_TECH_ID_LIST = AVNO

TC_DIAG_OUTPUT_DIR = {OUTPUT_BASE}/tc_diag
TC_DIAG_OUTPUT_TEMPLATE = {date?fmt=%Y}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

BOTH_VAR1_NAME = PRMSL
BOTH_VAR1_LEVELS = Z0

BOTH_VAR2_NAME = TMP
BOTH_VAR2_LEVELS = P1000, P925, P850, P700, P500, P400, P300, P250, P200, P150, P100


###
# TCDiag Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcdiag
###

LOG_TC_DIAG_VERBOSITY = 2

TC_DIAG_CONFIG_FILE = {PARM_BASE}/met_config/TCDiagConfig_wrapped

MODEL = GFSO

TC_DIAG_STORM_ID = AL032023
TC_DIAG_BASIN = AL
TC_DIAG_CYCLONE = 03

TC_DIAG_INIT_INCLUDE = {init?fmt=%Y%m%d%H}
#TC_DIAG_VALID_BEG =
```

(continues on next page)

(continued from previous page)

```
#TC_DIAG_VALID_END =
#TC_DIAG_VALID_INCLUDE_LIST =
#TC_DIAG_VALID_EXCLUDE_LIST =
#TC_DIAG_VALID_HOUR_LIST =
#TC_DIAG_LEAD =

#TC_DIAG_DIAG_SCRIPT =

TC_DIAG_DOMAIN_INFO1_DOMAIN = parent
TC_DIAG_DOMAIN_INFO1_N_RANGE = 150
TC_DIAG_DOMAIN_INFO1_N_AZIMUTH = 8
TC_DIAG_DOMAIN_INFO1_DELTA_RANGE_KM = 10.0
#TC_DIAG_DOMAIN_INFO1_DIAG_SCRIPT =

TC_DIAG_DOMAIN_INFO2_DOMAIN = nest
TC_DIAG_DOMAIN_INFO2_N_RANGE = 150
TC_DIAG_DOMAIN_INFO2_N_AZIMUTH = 8
TC_DIAG_DOMAIN_INFO2_DELTA_RANGE_KM = 2.0


#TC_DIAG_CENSOR_THRESH =
#TC_DIAG_CENSOR_VAL =
#TC_DIAG_CONVERT =

#TC_DIAG_DATA_DOMAIN =
#TC_DIAG_DATA_LEVEL =

#TC_DIAG_REGRID_METHOD = NEAREST
#TC_DIAG_REGRID_WIDTH = 1
#TC_DIAG_REGRID_VLD_THRESH = 0.5
#TC_DIAG_REGRID_SHAPE = SQUARE
#TC_DIAG_REGRID_CENSOR_THRESH =
#TC_DIAG_REGRID_CENSOR_VAL =
#TC_DIAG_REGRID_CONVERT =

#TC_DIAG_COMPUTE_TANGENTIAL_AND_RADIAL_WINDS =
#TC_DIAG_U_WIND_FIELD_NAME =
#TC_DIAG_V_WIND_FIELD_NAME =
#TC_DIAG_TANGENTIAL_VELOCITY_FIELD_NAME =
#TC_DIAG_TANGENTIAL_VELOCITY_LONG_FIELD_NAME =
#TC_DIAG_RADIAL_VELOCITY_FIELD_NAME =
#TC_DIAG_RADIAL_VELOCITY_LONG_FIELD_NAME =
#TC_DIAG_VORTEX_REMOVAL =
#TC_DIAG_NC_RNG_AZI_FLAG =
#TC_DIAG_NC_DIAG_FLAG =
```

(continues on next page)

(continued from previous page)

```
#TC_DIAG_CIRA_DIAG_FLAG =
#TC_DIAG_OUTPUT_PREFIX =

#LOG_LEVEL=DEBUG
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TCDiag MET Configuration* (page 254) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// TC-Diag configuration file.
//
// For additional information, please see the MET Users Guide.
//
////////////////////////////////////////////////////////////////////////////////

//
// Filter input track data lines.
//

//
// Model
//
//model =
${METPLUS_MODEL}


//
// Storm identifier
//
//storm_id =
${METPLUS_STORM_ID}
```

(continues on next page)

(continued from previous page)

```
//
// Basin
//
//basin =
${METPLUS_BASIN}


//
// Cyclone number
//
//cyclone =
${METPLUS_CYCLONE}


//
// Model initialization time
//
//init_inc =
${METPLUS_INIT_INCLUDE}


//
// Subset by the valid time
//
//valid_beg =
${METPLUS_VALID_BEG}
//valid_end =
${METPLUS_VALID_END}

//valid_inc =
${METPLUS_VALID_INCLUDE_LIST}

//valid_exc =
${METPLUS_VALID_EXCLUDE_LIST}


//
// Subset by the valid hour and lead time.
//
//valid_hour =
${METPLUS_VALID_HOUR_LIST}

//lead =
${METPLUS_LEAD_LIST}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Python diagnostic scripts to be run
// May be set separately in each "domain_info" entry
//
//diag_script =
${METPLUS_DIAG_SCRIPT}

//
// Domain-specific cylindrical coordinate transformation
//
//domain_info = {
${METPLUS_DOMAIN_INFO_LIST}

////////////////////////////////////////////////////////////////////////////////

//
// Data censoring and conversion
// May be set separately in each diag_data "field" entry
//
// censor_thresh = [];
${METPLUS_CENSOR_THRESH}

// censor_val    = [];
${METPLUS_CENSOR_VAL}

// convert(x)    = x;
${METPLUS_CONVERT}


//
// Data fields
//
data = {

   ${METPLUS_DATA_FILE_TYPE}

   // If empty, the field is processed for all domains
   //domain = [];
   ${METPLUS_DATA_DOMAIN}

   // Pressure levels to be used, unless overridden below
```

(continues on next page)

(continued from previous page)

```
   //level =
   ${METPLUS_DATA_LEVEL}

   //field = [
   ${METPLUS_DATA_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Regridding options
//
//regrid = {
${METPLUS_REGRID_DICT}

//
// Optionally convert u/v winds to tangential/radial winds
//
//compute_tangential_and_radial_winds =
${METPLUS_COMPUTE_TANGENTIAL_AND_RADIAL_WINDS}

//u_wind_field_name =
${METPLUS_U_WIND_FIELD_NAME}

//v_wind_field_name =
${METPLUS_V_WIND_FIELD_NAME}

//tangential_velocity_field_name =
${METPLUS_TANGENTIAL_VELOCITY_FIELD_NAME}

//tangential_velocity_long_field_name =
${METPLUS_TANGENTIAL_VELOCITY_LONG_FIELD_NAME}

//radial_velocity_field_name =
${METPLUS_RADIAL_VELOCITY_FIELD_NAME}

//radial_velocity_long_field_name =
${METPLUS_RADIAL_VELOCITY_LONG_FIELD_NAME}


//
// Vortex removal flag
//
//vortex_removal =
${METPLUS_VORTEX_REMOVAL}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Flags to control output files
//
//nc_rng_azi_flag =
${METPLUS_NC_RNG_AZI_FLAG}

//nc_diag_flag =
${METPLUS_NC_DIAG_FLAG}

//cira_diag_flag =
${METPLUS_CIRA_DIAG_FLAG}


////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

//output_prefix =
${METPLUS_OUTPUT_PREFIX}

//version       = "V11.1.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

Pass the use case configuration file to the run_metplus.py script along with any user-specific system configuration files if desired:

```
run_metplus.py /path/to/METplus/parm/use_cases/met_tool_wrapper/TCDiag/TCDiag.conf /path/to/
↪user_system.conf
```

See *Running METplus* (page 26) for more information.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/TCDiag (relative to **OUTPUT_BASE**) and will contain the following files:

- tc_diag_AL032023_GFSO_2023062012_cyl_grid_nest.nc
- tc_diag_AL032023_GFSO_2023062012_cyl_grid_parent.nc

Keywords

Note:

- DiagnosticsUseCase
- TCDiagToolUseCase
- GRIB2FileUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-TCDiag.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.27 TCGen

7.1.32.27.1 TCGen: Basic Use Case

met_tool_wrapper/TCGen/TCGen.conf

Scientific Objective

The TC-Gen tool provides verification of tropical cyclone genesis forecasts in ATCF file format.

Datasets

Track: A Deck or B Deck (Best)
Genesis: Genesis Forecast

Location: All of the input data required for this use case can be found in the met_tool_wrapper sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 711) section for more information.

METplus Components

This use case utilizes the METplus TCGen wrapper to search for files that match wildcard expressions and generate a command to run the MET tool tc_gen.

METplus Workflow

TCGen is the only tool called in this example. It processes the following run times:

Init: 2016

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c /path/to/TCGen.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/TCGen/TCGen.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###
```

(continues on next page)

(continued from previous page)

```
PROCESS_LIST = TCGen


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT

INIT_TIME_FMT = %Y
INIT_BEG = 2016

LEAD_SEQ =

TC_GEN_CUSTOM_LOOP_LIST =


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

TC_GEN_TRACK_INPUT_DIR = {INPUT_BASE}/met_test/tc_data/genesis/atcf
TC_GEN_TRACK_INPUT_TEMPLATE = {init?fmt=%Y}/*.dat

TC_GEN_GENESIS_INPUT_DIR = {INPUT_BASE}/met_test/tc_data/genesis/suite1
TC_GEN_GENESIS_INPUT_TEMPLATE = {init?fmt=%Y}*/genesis*{init?fmt=%Y}*

#TC_GEN_EDECK_INPUT_DIR =
#TC_GEN_EDECK_INPUT_TEMPLATE =

#TC_GEN_SHAPE_INPUT_DIR =
#TC_GEN_SHAPE_INPUT_TEMPLATE =

TC_GEN_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/TCGen
```

(continues on next page)

(continued from previous page)

```
TC_GEN_OUTPUT_TEMPLATE = tc_gen_{init?fmt=%Y}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###


###
# TCGen Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcgen
###

#LOG_TC_GEN_VERBOSITY = 2

TC_GEN_CONFIG_FILE = {PARM_BASE}/met_config/TCGenConfig_wrapped

TC_GEN_INIT_FREQ = 6

TC_GEN_VALID_FREQ = 6

TC_GEN_FCST_HR_WINDOW_BEGIN = 6
TC_GEN_FCST_HR_WINDOW_END = 120

TC_GEN_MIN_DURATION = 12

TC_GEN_FCST_GENESIS_VMAX_THRESH = NA
TC_GEN_FCST_GENESIS_MSLP_THRESH = NA

TC_GEN_BEST_GENESIS_TECHNIQUE = BEST
TC_GEN_BEST_GENESIS_CATEGORY = TD, TS
TC_GEN_BEST_GENESIS_VMAX_THRESH = NA
TC_GEN_BEST_GENESIS_MSLP_THRESH = NA

TC_GEN_OPER_TECHNIQUE = CARQ

# TC_GEN_FILTER_<n> sets filter items in the MET configuration file
# quotation marks within quotation marks must be preceeded with \
TC_GEN_FILTER_1 = desc = "AL_BASIN"; vx_mask = "MET_BASE/tc_data/basin_global_tenth_degree.
↪nc { name=\"basin\"; level=\"(*,*)\"; } ==1";
TC_GEN_FILTER_2 = desc = "AL_DLAND_300"; vx_mask = "MET_BASE/tc_data/basin_global_tenth_
↪degree.nc { name=\"basin\"; level=\"(*,*)\"; } ==1"; dland_thresh = >0&&<300;
TC_GEN_FILTER_3 = desc = "EP_CP_BASIN"; vx_mask = "MET_BASE/tc_data/basin_global_tenth_
↪degree.nc { name=\"basin\"; level=\"(*,*)\";} ==2||==3";
```

(continues on next page)

(continued from previous page)

```
TC_GEN_FILTER_4 = desc = "EP_BASIN"; genesis_window = { beg = -3*24; end = 3*24; }; genesis_
↪radius = 300;
TC_GEN_FILTER_5 = desc = "3DAY_300KM"; genesis_window = { beg = -3*24; end = 3*24; };␣
↪genesis_radius = 300;
TC_GEN_FILTER_6 = desc = "3DAY_600KM"; genesis_window = { beg = -3*24; end = 3*24; };␣
↪genesis_radius = 600;
TC_GEN_FILTER_7 = desc = "5DAY_300KM"; genesis_window = { beg = -5*24; end = 5*24; };␣
↪genesis_radius = 300;
TC_GEN_FILTER_8 = desc = "5DAY_600KM"; genesis_window = { beg = -5*24; end = 5*24; };␣
↪genesis_radius = 600;

TC_GEN_DESC = ALL

MODEL =

TC_GEN_STORM_ID =

TC_GEN_STORM_NAME =

TC_GEN_INIT_BEG =
TC_GEN_INIT_END =
TC_GEN_INIT_INC =
TC_GEN_INIT_EXC =

TC_GEN_VALID_BEG =
TC_GEN_VALID_END =

TC_GEN_INIT_HOUR =

TC_GEN_VX_MASK =

TC_GEN_BASIN_MASK =

TC_GEN_DLAND_THRESH = NA

TC_GEN_GENESIS_MATCH_RADIUS = 500

#TC_GEN_GENESIS_MATCH_POINT_TO_TRACK = True

#TC_GEN_GENESIS_MATCH_WINDOW_BEG = 0
#TC_GEN_GENESIS_MATCH_WINDOW_END = 0

#TC_GEN_OPS_HIT_WINDOW_BEG = 0
#TC_GEN_OPS_HIT_WINDOW_END = 48
```

(continues on next page)

(continued from previous page)

```
TC_GEN_DEV_HIT_RADIUS = 500

TC_GEN_DEV_HIT_WINDOW_BEGIN = -24
TC_GEN_DEV_HIT_WINDOW_END = 24

TC_GEN_DISCARD_INIT_POST_GENESIS_FLAG = True

TC_GEN_DEV_METHOD_FLAG = True

TC_GEN_OPS_METHOD_FLAG = True

TC_GEN_CI_ALPHA = 0.05

TC_GEN_OUTPUT_FLAG_FHO = NONE
TC_GEN_OUTPUT_FLAG_CTC = BOTH
TC_GEN_OUTPUT_FLAG_CTS = BOTH
TC_GEN_OUTPUT_FLAG_PCT = NONE
TC_GEN_OUTPUT_FLAG_PSTD = NONE
TC_GEN_OUTPUT_FLAG_PJC = NONE
TC_GEN_OUTPUT_FLAG_PRC = NONE
TC_GEN_OUTPUT_FLAG_GENMPR = NONE


TC_GEN_NC_PAIRS_FLAG_LATLON = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_GENESIS = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_TRACKS  = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_FY_OY   = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_FY_ON   = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_GENESIS = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_TRACKS  = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_FY_OY   = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_FN_OY   = TRUE

TC_GEN_VALID_MINUS_GENESIS_DIFF_THRESH = NA

TC_GEN_BEST_UNIQUE_FLAG = TRUE

TC_GEN_DLAND_FILE = MET_BASE/tc_data/dland_global_tenth_degree.nc

TC_GEN_BASIN_FILE = MET_BASE/tc_data/basin_global_tenth_degree.nc

TC_GEN_NC_PAIRS_GRID = G003
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TCGen MET Configuration* (page 266) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// TC-Gen configuration file.
//
// For additional information, see the MET_BASE/config/README_TC file.
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

////////////////////////////////////////////////////////////////////////////////
//
// Genesis event definition criteria.
//
////////////////////////////////////////////////////////////////////////////////

//
// Model initialization frequency in hours, starting at 0.
//
// init_freq =
${METPLUS_INIT_FREQ}

//
// Valid hour frequency to be analyzed in hours, starting at 0
//
// valid_freq =
${METPLUS_VALID_FREQ}

//
```

(continues on next page)

(continued from previous page)

```
// Forecast hours to be searched for genesis events
//
// fcst_hr_window =
${METPLUS_FCST_HR_WINDOW_DICT}

//
// Minimum track duration for genesis event in hours.
//
// min_duration =
${METPLUS_MIN_DURATION}

//
// Forecast genesis event criteria.  Defined as tracks reaching the specified
// intensity category, maximum wind speed threshold, and minimum sea-level
// pressure threshold.  The forecast genesis time is the valid time of the first
// track point where all of these criteria are met.
//
// fcst_genesis =
${METPLUS_FCST_GENESIS_DICT}

//
// BEST track genesis event criteria.  Defined as tracks reaching the specified
// intensity category, maximum wind speed threshold, and minimum sea-level
// pressure threshold.  The BEST track genesis time is the valid time of the
// first track point where all of these criteria are met.
//
// best_genesis =
${METPLUS_BEST_GENESIS_DICT}

//
// Operational track technique name
//
// oper_technique =
${METPLUS_OPER_TECHNIQUE}

////////////////////////////////////////////////////////////////////////////////
//
// Track filtering options
// May be specified separately in each filter array entry.
//
////////////////////////////////////////////////////////////////////////////////

//
// Array of dictionaries containing the track filtering options
// If empty, a single filter is defined using the top-level settings.
```

(continues on next page)

(continued from previous page)

```
//
// filter =
${METPLUS_FILTER}

//
// Description written to output DESC column
//
// desc =
${METPLUS_DESC}

//
// Forecast ATCF ID's
// If empty, all ATCF ID's found will be processed.
// Statistics will be generated separately for each ATCF ID.
//
// model =
${METPLUS_MODEL}

//
// BEST and operational track storm identifiers
//
// storm_id =
${METPLUS_STORM_ID}

//
// BEST and operational track storm names
//
// storm_name =
${METPLUS_STORM_NAME}

//
// Forecast and operational initialization times to include or exclude
//
// init_beg =
${METPLUS_INIT_BEG}

// init_end =
${METPLUS_INIT_END}

// init_inc =
${METPLUS_INIT_INC}

// init_exc =
${METPLUS_INIT_EXC}
```

(continues on next page)

(continued from previous page)

```
//
// Forecast, BEST, and operational valid time window
//
// valid_beg =
${METPLUS_VALID_BEG}

// valid_end =
${METPLUS_VALID_END}

//
// Forecast and operational initialization hours
//
// init_hour =
${METPLUS_INIT_HOUR}

//
// Forecast and operational lead times in hours
//
// lead =
${METPLUS_LEAD}

//
// Spatial masking region (path to gridded data file or polyline file)
//
// vx_mask =
${METPLUS_VX_MASK}

//
// Spatial masking of hurricane basin names from the basin_file
//
// basin_mask =
${METPLUS_BASIN_MASK}

//
// Distance to land threshold
//
//dland_thresh =
${METPLUS_DLAND_THRESH}

////////////////////////////////////////////////////////////////////////////////
//
// Matching and scoring options
// May be specified separately in each filter array entry.
//
////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Genesis matching logic. Compare the forecast genesis point to all points in
// the Best track (TRUE) or the single Best track genesis point (FALSE).
//
//genesis_match_point_to_track =
${METPLUS_GENESIS_MATCH_POINT_TO_TRACK}

//
// Radius in km to search for a matching genesis event
//
// genesis_match_radius =
${METPLUS_GENESIS_MATCH_RADIUS}

//
// Time window in hours, relative to the model genesis time, to search for a
// matching Best track point
//
//genesis_match_window = {
${METPLUS_GENESIS_MATCH_WINDOW_DICT}

//
// Radius in km for a development scoring method hit
//
// dev_hit_radius =
${METPLUS_DEV_HIT_RADIUS}

//
// Time window in hours for a development scoring method hit
//
// dev_hit_window =
${METPLUS_DEV_HIT_WINDOW_DICT}

// Time window in hours for the Best track genesis minus model initialization
// time difference for an operational scoring method hit
//
//ops_hit_window = {
${METPLUS_OPS_HIT_WINDOW_DICT}

//
// Discard genesis forecasts for initializations at or after the matching
// BEST track genesis time
//
// discard_init_post_genesis_flag =
${METPLUS_DISCARD_INIT_POST_GENESIS_FLAG}
```

(continues on next page)

(continued from previous page)

```
//
// Scoring methods to be applied
//
//dev_method_flag =
${METPLUS_DEV_METHOD_FLAG}

// ops_method_flag =
${METPLUS_OPS_METHOD_FLAG}

////////////////////////////////////////////////////////////////////////////////
//
// Output options
// May be specified separately in each filter array entry.
//
////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval alpha value
//
// ci_alpha =
${METPLUS_CI_ALPHA}

//
// Statistical output types
//
// output_flag =
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF genesis pair counts
//
// nc_pairs_flag =
${METPLUS_NC_PAIRS_FLAG_DICT}

//
// Specify which track points should be counted by thresholding the track point
// valid time minus genesis time difference.
//
// valid_minus_genesis_diff_thresh =
${METPLUS_VALID_MINUS_GENESIS_DIFF_THRESH}

//
// Count unique BEST track genesis event locations (TRUE) versus counting the
// location for all pairs (FALSE).
```

(continues on next page)

(continued from previous page)

```
//
// best_unique_flag =
${METPLUS_BEST_UNIQUE_FLAG}

////////////////////////////////////////////////////////////////////////////////
//
// Global settings
// May only be specified once.
//
////////////////////////////////////////////////////////////////////////////////

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
// dland_file =
${METPLUS_DLAND_FILE}

//
// Specify the NetCDF file containing a gridded representation of the
// global basins.
//
// basin_file =
${METPLUS_BASIN_FILE}

//
// NetCDF genesis pairs grid
//
// nc_pairs_grid =
${METPLUS_NC_PAIRS_GRID}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V10.0.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways: 1) Passing in TCGen.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/TCGen/TCGen.conf -c /path/
↪to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in TCGen.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/TCGen/TCGen.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/TCGen and will contain the following files:

- tc_gen_2016_ctc.txt
- tc_gen_2016_cts.txt
- tc_gen_2016.stat

Keywords

Note:

- TCGenToolUseCase
- DTCOrgUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-TCGen.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.28 TCMPRPlotter

7.1.32.28.1 TCMPRPlotter: Basic Use Case

met_tool_wrapper/TCMPRPlotter/TCMPRPlotter.conf

Scientific Objective

Generate plots of tropical cyclone tracks.

Datasets

No datasets are used in this use case, the tc-pairs output from the MET tc-pairs tool is used as input.

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 715) section for more information.

METplus Components

This use case utilizes the METplus TCMPRPlotter wrapper to invoke the the MET script tcmpr_plotter.R.

METplus Workflow

tcmpr_plotter.R is the only tool (script) called in this example.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/TCMPRPlotter/TCMPRPlotter.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/TCMPRPlotter/
↪TCMPRPlotter.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCMPRPlotter


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m
```

(continues on next page)

(continued from previous page)

```
INIT_BEG = 201503
INIT_END = 201503
INIT_INCREMENT = 6H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

TCMPR_PLOTTER_TCMPR_DATA_DIR = {INPUT_BASE}/met_test/tc_pairs/{date?fmt=%Y%m}
TCMPR_PLOTTER_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/tcmpr_plots


###
# TCMPRPlotter Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcmprplotter
###

#TCMPR_PLOTTER_READ_ALL_FILES = True

TCMPR_PLOTTER_CONFIG_FILE = {PARM_BASE}/use_cases/met_tool_wrapper/TCMPRPlotter/
↪TCMPRPlotterConfig_Customize
TCMPR_PLOTTER_PREFIX =
TCMPR_PLOTTER_TITLE =
TCMPR_PLOTTER_SUBTITLE = Your Subtitle Goes Here
TCMPR_PLOTTER_XLAB =
TCMPR_PLOTTER_YLAB = Your y-label Goes Here
TCMPR_PLOTTER_XLIM =
TCMPR_PLOTTER_YLIM =
TCMPR_PLOTTER_FILTER =

TCMPR_PLOTTER_FILTERED_TCST_DATA_FILE =

TCMPR_PLOTTER_DEP_VARS = AMSLP-BMSLP,AMAX_WIND-BMAX_WIND,TK_ERR
TCMPR_PLOTTER_SCATTER_X =
TCMPR_PLOTTER_SCATTER_Y =
TCMPR_PLOTTER_SKILL_REF =
TCMPR_PLOTTER_SERIES =
TCMPR_PLOTTER_SERIES_CI =
TCMPR_PLOTTER_LEGEND =
TCMPR_PLOTTER_LEAD =

TCMPR_PLOTTER_PLOT_TYPES = MEAN, MEDIAN
```

(continues on next page)

(continued from previous page)

```
TCMPR_PLOTTER_RP_DIFF =
TCMPR_PLOTTER_DEMO_YR =
TCMPR_PLOTTER_HFIP_BASELINE =
TCMPR_PLOTTER_FOOTNOTE_FLAG =
TCMPR_PLOTTER_PLOT_CONFIG_OPTS =
TCMPR_PLOTTER_SAVE_DATA =

TCMPR_PLOTTER_NO_EE = no
TCMPR_PLOTTER_NO_LOG = no
TCMPR_PLOTTER_SAVE = no
```

MET Configuration

A MET configuration is not needed to run this single wrapper use case.

Running METplus

This use case can be run two ways:

1) Passing in TCMPRPlotter.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/TCMPRPlotter/
↪TCMPRPlotter.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in TCMPRPlotter.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/TCMPRPlotter/
↪TCMPRPlotter.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in tcmpr_plots/2005080700 (relative to **OUTPUT_BASE**) and will contain the following files:

- AMAX_WIND-BMAX_WIND_mean.png
- AMAX_WIND-BMAX_WIND_mean.png
- AMSLP-BMSLP_mean.png
- AMSLP-BMSLP_median.png
- TK_ERR_mean.png
- TK_ERR_median.png

Keywords

Note:

- TCMPRPlotterUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.29 TCPairs

7.1.32.29.1 TCPairs: Basic Use Case for Extra Tropical Cyclones

met_tool_wrapper/TCPairs/TCPairs_extra_tropical.conf

Scientific Objective

Once this method is complete, a forecast and reference track analysis file will have been paired up, allowing statistical information to be extracted.

Datasets

Forecast: A Deck
track_data/201412/a??q201412*.gfso.*
Observation: Best Track - B Deck
track_data/201412/b??q201412*.gfso.*

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 725) section for more information.

Data Source: GFS

METplus Components

This use case utilizes the METplus TCPairs wrapper to search for files that are valid at a given run time and generate a command to run the MET tool tc_pairs.

METplus Workflow

TCPairs is the only tool called in this example. It processes the following run times:

Init: 2014-12-13_18Z
Forecast lead: 24 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c /path/to/TCPairs_extra_tropical.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/TCPairs/TCPairs_extra_
↪tropical.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCPairs


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2014121318
INIT_END = 2014121318
INIT_INCREMENT = 21600

TC_PAIRS_RUN_ONCE = True


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/met_test/new/track_data
TC_PAIRS_ADECK_TEMPLATE = {date?fmt=%Y%m}/a{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}

TC_PAIRS_BDECK_INPUT_DIR = {TC_PAIRS_ADECK_INPUT_DIR}
TC_PAIRS_BDECK_TEMPLATE = {date?fmt=%Y%m}/b{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}

TC_PAIRS_REFORMAT_DIR = {OUTPUT_BASE}/track_data_atcf

#TC_PAIRS_DIAG_DIR1 =
#TC_PAIRS_DIAG_TEMPLATE1 =
#TC_PAIRS_DIAG_SOURCE1 =

TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs
TC_PAIRS_OUTPUT_TEMPLATE = {date?fmt=%Y%m}/{basin?fmt=%s}q{date?fmt=%Y%m%d%H}.gfso.{cyclone?
↪fmt=%s}


TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = yes
TC_PAIRS_SKIP_IF_REFORMAT_EXISTS = yes

TC_PAIRS_READ_ALL_FILES = no
#TC_PAIRS_SKIP_LEAD_SEQ = False

TC_PAIRS_REFORMAT_DECK = yes
TC_PAIRS_REFORMAT_TYPE = SBU


###
# TCPairs Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcpairs
###

TC_PAIRS_CONFIG_FILE = {PARM_BASE}/met_config/TCPairsConfig_wrapped

TC_PAIRS_INIT_INCLUDE =
TC_PAIRS_INIT_EXCLUDE =

TC_PAIRS_INIT_BEG = 2014121318
TC_PAIRS_INIT_END = 2014121418
```

(continues on next page)

(continued from previous page)

```
#TC_PAIRS_VALID_INCLUDE =
#TC_PAIRS_VALID_EXCLUDE =

#TC_PAIRS_WRITE_VALID =

TC_PAIRS_VALID_BEG =
TC_PAIRS_VALID_END =

MODEL =

#TC_PAIRS_DESC =

TC_PAIRS_STORM_ID =
TC_PAIRS_BASIN =
TC_PAIRS_CYCLONE =
TC_PAIRS_STORM_NAME =

TC_PAIRS_DLAND_FILE = {MET_INSTALL_DIR}/share/met/tc_data/dland_global_tenth_degree.nc

TC_PAIRS_MISSING_VAL_TO_REPLACE = -99
TC_PAIRS_MISSING_VAL = -9999

#TC_PAIRS_CONSENSUS1_NAME =
#TC_PAIRS_CONSENSUS1_MEMBERS =
#TC_PAIRS_CONSENSUS1_REQUIRED =
#TC_PAIRS_CONSENSUS1_MIN_REQ =
#TC_PAIRS_CONSENSUS1_WRITE_MEMBERS =

#TC_PAIRS_CHECK_DUP =

#TC_PAIRS_INTERP12 =

#TC_PAIRS_MATCH_POINTS =

#TC_PAIRS_DIAG_INFO_MAP1_DIAG_SOURCE =
#TC_PAIRS_DIAG_INFO_MAP1_TRACK_SOURCE =
#TC_PAIRS_DIAG_INFO_MAP1_FIELD_SOURCE =
#TC_PAIRS_DIAG_INFO_MAP1_MATCH_TO_TRACK =
#TC_PAIRS_DIAG_INFO_MAP1_DIAG_NAME =

#TC_PAIRS_DIAG_CONVERT_MAP1_DIAG_SOURCE =
#TC_PAIRS_DIAG_CONVERT_MAP1_KEY =
#TC_PAIRS_DIAG_CONVERT_MAP1_CONVERT =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TCPairs MET Configuration* (page 282) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

//
// Models
//
${METPLUS_MODEL}

//
// Description
//
${METPLUS_DESC}

//
// Storm identifiers
//
${METPLUS_STORM_ID}

//
// Basins
//
${METPLUS_BASIN}

//
```

(continues on next page)

(continued from previous page)

```
// Cyclone numbers
//
${METPLUS_CYCLONE}

//
// Storm names
//
${METPLUS_STORM_NAME}

//
// Model initialization time windows to include or exclude
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
${METPLUS_VALID_INC}
// valid_exc =
${METPLUS_VALID_EXC}

// write_valid =
${METPLUS_WRITE_VALID}

//
// Valid model time window
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}

//
// Model initialization hours
//
init_hour = [];

//
// Required lead time in hours
//
lead_req = [];

//
// lat/lon polylines defining masking regions
```

(continues on next page)

(continued from previous page)

```
//
init_mask  = "";
valid_mask = "";

//
// Specify if the code should check for duplicate ATCF lines
//
//check_dup =
${METPLUS_CHECK_DUP}


//
// Specify special processing to be performed for interpolated models.
// Set to NONE, FILL, or REPLACE.
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
//
//consensus =
${METPLUS_CONSENSUS_LIST}


//
// Forecast lag times
//
lag_time = [];

//
// CLIPER/SHIFOR baseline forecasts to be derived from the BEST
// and operational (CARQ) tracks.
//
best_technique = [ "BEST" ];
best_baseline  = [];
oper_technique = [ "CARQ" ];
oper_baseline  = [];

//
// Specify the datasets to be searched for analysis tracks (NONE, ADECK, BDECK,
// or BOTH).
//
anly_track = BDECK;
```

(continues on next page)

(continued from previous page)

```
//
// Specify if only those track points common to both the ADECK and BDECK
// tracks be written out.
//
//match_points =
${METPLUS_MATCH_POINTS}

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
${METPLUS_DLAND_FILE}

//
// Specify watch/warning information:
//   - Input watch/warning filename
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}


//diag_info_map = {
${METPLUS_DIAG_INFO_MAP_LIST}

//diag_convert_map = {
${METPLUS_DIAG_CONVERT_MAP_LIST}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

It is recommended to run this use case by:

Passing in TCPairs_extra_tropical.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/TCPairs_extra_tropical.conf -c /path/to/user_system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in **tc_pairs/201412** (relative to **OUTPUT_BASE**) and will contain the following files:

- mlq2014121318.gfso.0103.tcst
- mlq2014121318.gfso.0104.tcst

Keywords

Note:

- TCPairsToolUseCase
- SBUOrgUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-TCPairs.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.29.2 TCPairs: Basic Use Case for Tropical Cyclones

met_tool_wrapper/TCPairs/TCPairs_tropical.conf

Scientific Objective

Once this method is complete, a forecast and reference track analysis file will have been paired up, allowing statistical information to be extracted.

Datasets

Forecast: A Deck
/path/to/hwrf/adeck
Observation: Best Track - B Deck
/path/to/hwrf/bdeck

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 734) section for more information.

Data Source: HWRF

METplus Components

This use case utilizes the METplus TCPairs wrapper to search for files that are valid at a given run time and generate a command to run the MET tool tc_pairs.

METplus Workflow

TCPairs is the only tool called in this example. It processes the following run times:

Init: 2018-08-30_06Z, 2018-08-30_12Z, 2018-08-30_18Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c /path/to/TCPairs_tropical.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/TCPairs/TCPairs_
↪tropical.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCPairs


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2018083006
```

(continues on next page)

(continued from previous page)

```
INIT_END = 2018083018
INIT_INCREMENT = 21600

#TC_PAIRS_SKIP_LEAD_SEQ = False

TC_PAIRS_RUN_ONCE = False


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/met_test/new/hwrf/adeck
TC_PAIRS_ADECK_TEMPLATE = {model?fmt=%s}/*{cyclone?fmt=%s}l.{date?fmt=%Y%m%d%H}.trak.hwrf.
↪atcfunix

TC_PAIRS_BDECK_INPUT_DIR = {INPUT_BASE}/met_test/new/hwrf/bdeck
TC_PAIRS_BDECK_TEMPLATE = b{basin?fmt=%s}{cyclone?fmt=%s}{date?fmt=%Y}.dat

TC_PAIRS_EDECK_INPUT_DIR =
TC_PAIRS_EDECK_TEMPLATE =

#TC_PAIRS_DIAG_DIR1 =
#TC_PAIRS_DIAG_TEMPLATE1 =
#TC_PAIRS_DIAG_SOURCE1 =

TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs
TC_PAIRS_OUTPUT_TEMPLATE = tc_pairs_{basin?fmt=%s}{date?fmt=%Y%m%d%H}.dat

TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = no
TC_PAIRS_READ_ALL_FILES = no
TC_PAIRS_REFORMAT_DECK = no


###
# TCPairs Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcpairs
###

TC_PAIRS_CONFIG_FILE = {PARM_BASE}/met_config/TCPairsConfig_wrapped

TC_PAIRS_INIT_INCLUDE =
TC_PAIRS_INIT_EXCLUDE =
```

(continues on next page)

(continued from previous page)

```
TC_PAIRS_INIT_BEG =
TC_PAIRS_INIT_END =

#TC_PAIRS_VALID_INCLUDE =
#TC_PAIRS_VALID_EXCLUDE =

#TC_PAIRS_WRITE_VALID =

TC_PAIRS_VALID_BEG =
TC_PAIRS_VALID_END =

MODEL = MYNN, H19C, H19M, CTRL, MYGF

#TC_PAIRS_DESC =

#TC_PAIRS_STORM_ID = al062018, al092018, al132018, al142018
#TC_PAIRS_BASIN = AL
TC_PAIRS_CYCLONE = 06
TC_PAIRS_STORM_NAME =

TC_PAIRS_DLAND_FILE = MET_BASE/tc_data/dland_global_tenth_degree.nc

#TC_PAIRS_CONSENSUS1_NAME =
#TC_PAIRS_CONSENSUS1_MEMBERS =
#TC_PAIRS_CONSENSUS1_REQUIRED =
#TC_PAIRS_CONSENSUS1_MIN_REQ =
#TC_PAIRS_CONSENSUS1_WRITE_MEMBERS =

#TC_PAIRS_CHECK_DUP =

#TC_PAIRS_INTERP12 =

#TC_PAIRS_MATCH_POINTS =

#TC_PAIRS_DIAG_INFO_MAP1_DIAG_SOURCE =
#TC_PAIRS_DIAG_INFO_MAP1_TRACK_SOURCE =
#TC_PAIRS_DIAG_INFO_MAP1_FIELD_SOURCE =
#TC_PAIRS_DIAG_INFO_MAP1_MATCH_TO_TRACK =
#TC_PAIRS_DIAG_INFO_MAP1_DIAG_NAME =

#TC_PAIRS_DIAG_CONVERT_MAP1_DIAG_SOURCE =
#TC_PAIRS_DIAG_CONVERT_MAP1_KEY =
#TC_PAIRS_DIAG_CONVERT_MAP1_CONVERT =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TCPairs MET Configuration* (page 282) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

//
// Models
//
${METPLUS_MODEL}

//
// Description
//
${METPLUS_DESC}

//
// Storm identifiers
//
${METPLUS_STORM_ID}

//
// Basins
//
${METPLUS_BASIN}

//
```

(continues on next page)

(continued from previous page)

```
// Cyclone numbers
//
${METPLUS_CYCLONE}

//
// Storm names
//
${METPLUS_STORM_NAME}

//
// Model initialization time windows to include or exclude
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
${METPLUS_VALID_INC}
// valid_exc =
${METPLUS_VALID_EXC}

// write_valid =
${METPLUS_WRITE_VALID}

//
// Valid model time window
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}

//
// Model initialization hours
//
init_hour = [];

//
// Required lead time in hours
//
lead_req = [];

//
// lat/lon polylines defining masking regions
```

(continues on next page)

(continued from previous page)

```
//
init_mask  = "";
valid_mask = "";

//
// Specify if the code should check for duplicate ATCF lines
//
//check_dup =
${METPLUS_CHECK_DUP}


//
// Specify special processing to be performed for interpolated models.
// Set to NONE, FILL, or REPLACE.
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
//
//consensus =
${METPLUS_CONSENSUS_LIST}


//
// Forecast lag times
//
lag_time = [];

//
// CLIPER/SHIFOR baseline forecasts to be derived from the BEST
// and operational (CARQ) tracks.
//
best_technique = [ "BEST" ];
best_baseline  = [];
oper_technique = [ "CARQ" ];
oper_baseline  = [];

//
// Specify the datasets to be searched for analysis tracks (NONE, ADECK, BDECK,
// or BOTH).
//
anly_track = BDECK;
```

(continues on next page)

(continued from previous page)

```
//
// Specify if only those track points common to both the ADECK and BDECK
// tracks be written out.
//
//match_points =
${METPLUS_MATCH_POINTS}

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
${METPLUS_DLAND_FILE}

//
// Specify watch/warning information:
//   - Input watch/warning filename
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}


//diag_info_map = {
${METPLUS_DIAG_INFO_MAP_LIST}

//diag_convert_map = {
${METPLUS_DIAG_CONVERT_MAP_LIST}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in TCPairs_tropical.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/TCPairs_tropical.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in TCPairs_tropical.conf:

```
run_metplus.py -c /path/to/TCPairs_tropical.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in tc_pairs and will contain the following files:

- tc_pairs_al2018083006.dat.tcst
- tc_pairs_al2018083012.dat.tcst
- tc_pairs_al2018083018.dat.tcst

Keywords

Note:

- TCPairsToolUseCase
- DTCOrgUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-TCPairs.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.30 TCRMW

7.1.32.30.1 TCRMW: Basic Use Case

met_tool_wrapper/TCRMW/TCRMW.conf

Scientific Objective

The TC-RMW tool regrids tropical cyclone model data onto a moving range-azimuth grid centered on points along the storm track. This capability replicates the NOAA Hurricane Research Division DIA-Post module.

Datasets

Forecast: GFS FV3
Track: A Deck

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 740) section for more information.

METplus Components

This use case utilizes the METplus TCRMW wrapper to search for the desired ADECK file and forecast files that are correspond to the track. It generates a command to run the MET tool TC-RMW if all required files are found.

METplus Workflow

TCRMW is the only tool called in this example. It processes the following run times:

Init: 2016-09-29- 00Z
Forecast lead: 141, 143, and 147 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/TCRMW/TCRMW.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/TCRMW/TCRMW.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCRMW


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2016092900
INIT_END = 2016092900
INIT_INCREMENT = 21600


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

TC_RMW_DECK_INPUT_DIR = {INPUT_BASE}/met_test/new/tc_data/adeck
TC_RMW_DECK_TEMPLATE = aal14{date?fmt=%Y}_short.dat

TC_RMW_INPUT_DIR = {INPUT_BASE}/met_test/new/model_data/grib2/gfs_fv3
TC_RMW_INPUT_TEMPLATE = gfs.subset.t00z.pgrb2.0p25.f*

TC_RMW_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/TCRMW
TC_RMW_OUTPUT_TEMPLATE = tc_rmw_aal14{date?fmt=%Y}.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

BOTH_VAR1_NAME = PRMSL
BOTH_VAR1_LEVELS = L0

BOTH_VAR2_NAME = TMP
BOTH_VAR2_LEVELS = P1000, P900, P800, P700, P500, P100


###
# TCRMW Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcrmw
###
```

(continues on next page)

(continued from previous page)

```
#LOG_TC_RMW_VERBOSITY = 2

TC_RMW_CONFIG_FILE = {PARM_BASE}/met_config/TCRMWConfig_wrapped

MODEL = fv3

#TC_RMW_DESC =

#TC_RMW_REGRID_METHOD = NEAREST
#TC_RMW_REGRID_WIDTH = 1
#TC_RMW_REGRID_VLD_THRESH = 0.5
#TC_RMW_REGRID_SHAPE = SQUARE
#TC_RMW_REGRID_CONVERT =
#TC_RMW_REGRID_CENSOR_THRESH =
#TC_RMW_REGRID_CENSOR_VAL =

TC_RMW_STORM_ID = AL142016
TC_RMW_BASIN = AL
TC_RMW_CYCLONE = 14

#TC_RMW_N_RANGE = 100
#TC_RMW_N_AZIMUTH = 180
#TC_RMW_MAX_RANGE_KM = 1000.0
#TC_RMW_DELTA_RANGE_KM = 10.0
#TC_RMW_SCALE = 0.2

#TC_RMW_INIT_INCLUDE =
#TC_RMW_VALID_BEG =
#TC_RMW_VALID_END =
#TC_RMW_VALID_INCLUDE_LIST =
#TC_RMW_VALID_EXCLUDE_LIST =
#TC_RMW_VALID_HOUR_LIST =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TCRMW MET Configuration* (page 290) section of the User's Guide for more information on

the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// TC-RMW configuration file.
//
// For additional information, see the MET_BASE/config/README_TC file.
//
////////////////////////////////////////////////////////////////////////////////

// The following environment variables set the text if the corresponding
// variables at defined in the METplus config. If not, they are set to
// and empty string, which will cause MET to use the value defined in the
// default configuration file.

${METPLUS_MODEL}

${METPLUS_STORM_ID}
${METPLUS_BASIN}
${METPLUS_CYCLONE}
${METPLUS_INIT_INCLUDE}

${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INCLUDE_LIST}
${METPLUS_VALID_EXCLUDE_LIST}

${METPLUS_VALID_HOUR_LIST}
${METPLUS_LEAD_LIST}

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];

//
// Data fields
//
data = {
   ${METPLUS_DATA_FILE_TYPE}

   ${METPLUS_DATA_FIELD}
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Regridding options
//
${METPLUS_REGRID_DICT}

//
// Range-Azimuth grid parameters
//
// The following environmnet variables set the text if the corresponding
// variables at defined in the METplus config. If not, they are set to
// and empty string, which will cause MET to use the value defined in the
// default configuration file.

${METPLUS_N_RANGE}
${METPLUS_N_AZIMUTH}
${METPLUS_MAX_RANGE_KM}
${METPLUS_DELTA_RANGE_KM}
${METPLUS_RMW_SCALE}

////////////////////////////////////////////////////////////////////////////////

//version = "V10.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in TCRMW.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/TCRMW/TCRMW.conf -c /
↪path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in TCRMW.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/TCRMW/TCRMW.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_tool_wrapper/TCRMW (relative to **OUTPUT_BASE**) and will contain the following files:

- tc_rmw_aal142016.nc

Keywords

Note:

- TCRMWToolUseCase
- GRIB2FileUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-TCRMW.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.31 TCStat

7.1.32.31.1 TCStat: Basic Use Case

met_tool_wrapper/TCStat/TCStat.conf

Scientific Objective

Summarize and stratify the data from TC-Pairs track and intensity data

Datasets

TC-Pairs data from 201503

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 750) section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus TCStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool tc_stat if all required files are found.

METplus Workflow

TCStat is the only tool called in this example. It processes the following TCST run times:

TCST: 2015030100

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/TCStat/TCStat.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/TCStat/TCStat.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2019103112
INIT_END = 2019103112
INIT_INCREMENT = 6H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

```

(continues on next page)

(continued from previous page)

```
TC_STAT_LOOKIN_DIR = {INPUT_BASE}/met_test/tc_pairs

TC_STAT_OUTPUT_DIR = {OUTPUT_BASE}/tc_stat
TC_STAT_OUTPUT_TEMPLATE = job.out


###
# TCStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcstat
###

TC_STAT_CONFIG_FILE = {PARM_BASE}/met_config/TCStatConfig_wrapped

TC_STAT_JOB_ARGS = -job summary -line_type TCMPR -column 'ASPEED' -dump_row {TC_STAT_OUTPUT_
↪DIR}/tc_stat_summary.tcst

TC_STAT_AMODEL =
TC_STAT_BMODEL =
TC_STAT_DESC =
TC_STAT_STORM_ID =
TC_STAT_BASIN =
TC_STAT_CYCLONE =
TC_STAT_STORM_NAME =

TC_STAT_INIT_BEG = 20150301
TC_STAT_INIT_END = 20150304
TC_STAT_INIT_INCLUDE =
TC_STAT_INIT_EXCLUDE =
TC_STAT_INIT_HOUR = 00

TC_STAT_VALID_BEG =
TC_STAT_VALID_END =
TC_STAT_VALID_INCLUDE =
TC_STAT_VALID_EXCLUDE =
TC_STAT_VALID_HOUR =
TC_STAT_LEAD_REQ =
TC_STAT_INIT_MASK =
TC_STAT_VALID_MASK =

#TC_STAT_LINE_TYPE =

TC_STAT_LEAD =

TC_STAT_TRACK_WATCH_WARN =
```

(continues on next page)

(continued from previous page)

```
TC_STAT_COLUMN_THRESH_NAME =
TC_STAT_COLUMN_THRESH_VAL =

TC_STAT_COLUMN_STR_NAME =
TC_STAT_COLUMN_STR_VAL =

TC_STAT_INIT_THRESH_NAME =
TC_STAT_INIT_THRESH_VAL =

TC_STAT_INIT_STR_NAME =
TC_STAT_INIT_STR_VAL =

TC_STAT_WATER_ONLY =

TC_STAT_LANDFALL =

TC_STAT_LANDFALL_BEG = -24
TC_STAT_LANDFALL_END = 00

TC_STAT_MATCH_POINTS = false

#TC_STAT_COLUMN_STR_EXC_NAME =
#TC_STAT_COLUMN_STR_EXC_VAL =

#TC_STAT_INIT_STR_EXC_NAME =
#TC_STAT_INIT_STR_EXC_VAL =

#TC_STAT_DIAG_THRESH_NAME =
#TC_STAT_DIAG_THRESH_VAL =
#TC_STAT_INIT_DIAG_THRESH_NAME =
#TC_STAT_INIT_DIAG_THRESH_VAL =

#TC_STAT_EVENT_EQUAL =
#TC_STAT_EVENT_EQUAL_LEAD =
#TC_STAT_OUT_INIT_MASK =
#TC_STAT_OUT_VALID_MASK =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TCStat MET Configuration* (page 297) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCStat configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to filter the TC-STAT data down to the
// desired subset of lines over which statistics are to be computed.  Only
// those lines which meet ALL of the criteria specified will be retained.
//
// The settings that are common to all jobs may be specified once at the top
// level.  If no selection is listed for a parameter, that parameter will not
// be used for filtering.  If multiple selections are listed for a parameter,
// the analyses will be performed on their union.
//

//
// Stratify by the AMODEL or BMODEL columns.
//
${METPLUS_AMODEL}
${METPLUS_BMODEL}

//
// Stratify by the DESC column.
//
${METPLUS_DESC}

//
// Stratify by the STORM_ID column.
//
${METPLUS_STORM_ID}
```

(continues on next page)

(continued from previous page)

```
//
// Stratify by the BASIN column.
// May add using the "-basin" job command option.
//
${METPLUS_BASIN}

//
// Stratify by the CYCLONE column.
// May add using the "-cyclone" job command option.
//
${METPLUS_CYCLONE}

//
// Stratify by the STORM_NAME column.
// May add using the "-storm_name" job command option.
//
${METPLUS_STORM_NAME}

//
// Stratify by the INIT times.
// Model initialization time windows to include or exclude
// May modify using the "-init_beg", "-init_end", "-init_inc",
// and "-init_exc" job command options.
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
${METPLUS_INIT_INC}
${METPLUS_INIT_EXC}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INC}
${METPLUS_VALID_EXC}

//
// Stratify by the initialization and valid hours and lead time.
//
${METPLUS_INIT_HOUR}
${METPLUS_VALID_HOUR}
${METPLUS_LEAD}
```

(continues on next page)

(continued from previous page)

```
//
// Select tracks which contain all required lead times.
//
${METPLUS_LEAD_REQ}

//
// Stratify by the INIT_MASK and VALID_MASK columns.
//
${METPLUS_INIT_MASK}
${METPLUS_VALID_MASK}


//
// Stratify by the LINE_TYPE column.
//
//line_type =
${METPLUS_LINE_TYPE}


//
// Stratify by checking the watch/warning status for each track point
// common to both the ADECK and BDECK tracks.  If the watch/warning status
// of any of the track points appears in the list, retain the entire track.
//
${METPLUS_TRACK_WATCH_WARN}

//
// Stratify by applying thresholds to numeric data columns.
//
${METPLUS_COLUMN_THRESH_NAME}
${METPLUS_COLUMN_THRESH_VAL}

//
// Stratify by performing string matching on non-numeric data columns.
//
${METPLUS_COLUMN_STR_NAME}
${METPLUS_COLUMN_STR_VAL}

//
// Stratify by excluding strings in non-numeric data columns.
//
//column_str_exc_name =
${METPLUS_COLUMN_STR_EXC_NAME}

//column_str_exc_val =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_COLUMN_STR_EXC_VAL}

//
// Similar to the column_thresh options above
//
${METPLUS_INIT_THRESH_NAME}
${METPLUS_INIT_THRESH_VAL}

//
// Similar to the column_str options above
//
${METPLUS_INIT_STR_NAME}
${METPLUS_INIT_STR_VAL}

//
// Similar to the column_str_exc options above
//
//init_str_exc_name =
${METPLUS_INIT_STR_EXC_NAME}

//init_str_exc_val =
${METPLUS_INIT_STR_EXC_VAL}

//diag_thresh_name =
${METPLUS_DIAG_THRESH_NAME}

//diag_thresh_val =
${METPLUS_DIAG_THRESH_VAL}

//init_diag_thresh_name =
${METPLUS_INIT_DIAG_THRESH_NAME}

//init_diag_thresh_val =
${METPLUS_INIT_DIAG_THRESH_VAL}



//
// Stratify by the ADECK and BDECK distances to land.
//
${METPLUS_WATER_ONLY}

//
// Specify whether only those track points occurring near landfall should be
// retained, and define the landfall retention window in HH[MMSS] format
// around the landfall time.
```

(continues on next page)

(continued from previous page)

```
//
${METPLUS_LANDFALL}
${METPLUS_LANDFALL_BEG}
${METPLUS_LANDFALL_END}

//
// Specify whether only those track points common to both the ADECK and BDECK
// tracks should be retained.  May modify using the "-match_points" job command
// option.
//
${METPLUS_MATCH_POINTS}

//event_equal =
${METPLUS_EVENT_EQUAL}

//event_equal_lead =
${METPLUS_EVENT_EQUAL_LEAD}

//out_init_mask =
${METPLUS_OUT_INIT_MASK}

//out_valid_mask =
${METPLUS_OUT_VALID_MASK}


//
// Array of TCStat analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in TCStat.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/TCStat/TCStat.conf -
↪c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in TCStat.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/TCStat/TCStat.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
OUTPUT_BASE = /path/to/output/dir
INPUT_BASE/tc_pairs = path/to/tc_pairs/
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in tc_stat/201503 (relative to **OUTPUT_BASE**) and will contain the following files:

- tc_stat_summary.tcst

Keywords

Note:

- TCStatToolUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/met_tool_wrapper-TCStat.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.32 UserScript

7.1.32.32.1 UserScript: Run Once For Each Runtime Use Case

met_tool_wrapper/UserScript/UserScript_run_once_for_each.conf

Scientific Objective

Demonstrate how to run a user-defined script that should be executed once for each valid time and forecast lead combination. This use case runs a simple ls command to list the contents of a directory.

Datasets

Input: Empty test files from the METplus repository

METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command that is specified by the user.

METplus Workflow

UserScript is the only tool called in this example. It processes the following run times:

Valid: 2014-10-31 09:30:15
Forecast Lead: 0 hour
Forecast Lead: 12 hour
Forecast Lead: 24 hour
Forecast Lead: 120 hour

Valid: 2014-10-31 21:30:15
Forecast Lead: 0 hour
Forecast Lead: 12 hour
Forecast Lead: 24 hour
Forecast Lead: 120 hour

Valid: 2014-11-01 09:30:15

Forecast Lead: 0 hour
Forecast Lead: 12 hour
Forecast Lead: 24 hour
Forecast Lead: 120 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/UserScript/UserScript_run_once_for_each.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/UserScript/UserScript_
↪run_once_for_each.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_FOR_EACH

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H%M%S
```

(continues on next page)

(continued from previous page)

```
VALID_BEG = 20141031093015
VALID_END = 20141101093015
VALID_INCREMENT = 12H

LEAD_SEQ = 0H, 12H, 24H, 120H

USER_SCRIPT_CUSTOM_LOOP_LIST = nc


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

USER_SCRIPT_INPUT_TEMPLATE = init_{init?fmt=%Y%m%d%H%M%S}_valid_{valid?fmt=%Y%m%d%H%M%S}_
↪lead_{lead?fmt=%3H}.{custom}
USER_SCRIPT_INPUT_DIR = {INPUT_BASE}/met_test/new/test


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/met_tool_wrapper/UserScript/print_file_list.py
```

MET Configuration

None. UserScript does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in UserScript_run_once_for_each.conf_run_once then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/UserScript/
↪UserScript_run_once_for_each.conf_run_once -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_run_once_for_each.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/UserScript_
↪run_once_for_each.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

No output files are generated from this use case, but the logfile will contain the results of the directory listing command(s)

- init_20141031093015_valid_20141031093015_lead_000.nc
- init_20141030213015_valid_20141031093015_lead_012.nc
- init_20141030093015_valid_20141031093015_lead_024.nc
- init_20141026093015_valid_20141031093015_lead_120.nc
- init_20141031213015_valid_20141031213015_lead_000.nc
- init_20141031093015_valid_20141031213015_lead_012.nc
- init_20141030213015_valid_20141031213015_lead_024.nc
- init_20141026213015_valid_20141031213015_lead_120.nc
- init_20141101093015_valid_20141101093015_lead_000.nc
- init_20141031213015_valid_20141101093015_lead_012.nc
- init_20141031093015_valid_20141101093015_lead_024.nc
- init_20141027093015_valid_20141101093015_lead_120.nc

Keywords

Note:

- UserScriptUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.32.2 UserScript: Run Once Per Lead Use Case

met_tool_wrapper/UserScript/UserScript_run_once_per_lead.conf

Scientific Objective

Demonstrate how to run a user-defined script that should be executed once for each forecast lead time. This use case runs a simple ls command to list the contents of a directory. A wildcard character (*) is used to replace filename template tags for valid and init to find all files that match any of the times available.

Datasets

Input: Empty test files from the METplus repository

METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command that is specified by the user.

METplus Workflow

UserScript is the only tool called in this example. It processes the following run times:

Forecast Lead: 0 hour
Forecast Lead: 12 hour
Forecast Lead: 24 hour
Forecast Lead: 120 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/UserScript/UserScript_run_once_per_lead.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/UserScript/UserScript_
↪run_once_per_lead.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H%M%S
INIT_BEG = 20141031093015
INIT_END = 20141101093015
INIT_INCREMENT = 12H

LEAD_SEQ = 0H, 12H, 24H, 120H

USER_SCRIPT_CUSTOM_LOOP_LIST = nc
```

(continues on next page)

(continued from previous page)

```
###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

USER_SCRIPT_INPUT_TEMPLATE = init_{init?fmt=%Y%m%d%H%M%S}_valid_{valid?fmt=%Y%m%d%H%M%S}_
↪lead_{lead?fmt=%3H}.{custom}
USER_SCRIPT_INPUT_DIR = {INPUT_BASE}/met_test/new/test


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/met_tool_wrapper/UserScript/print_file_list.py
```

MET Configuration

None. UserScript does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in UserScript_run_once_per_lead.conf_run_once then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/UserScript/
↪UserScript_run_once_per_lead.conf_run_once -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_run_once_per_lead.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/UserScript_
↪run_once_per_lead.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases

- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

No output files are generated from this use case, but the logfile will contain the results of the directory listing command(s)

Forecast Lead: 0 hour

- init_20141031093015_valid_20141031093015_lead_000.nc
- init_20141031213015_valid_20141031213015_lead_000.nc
- init_20141101093015_valid_20141101093015_lead_000.nc

Forecast Lead: 12 hour

- init_20141030213015_valid_20141031093015_lead_012.nc
- init_20141031093015_valid_20141031213015_lead_012.nc
- init_20141031213015_valid_20141101093015_lead_012.nc
- init_20141101093015_valid_20141101213015_lead_012.nc

Forecast Lead: 24 hour

- init_20141030093015_valid_20141031093015_lead_024.nc
- init_20141030213015_valid_20141031213015_lead_024.nc
- init_20141031093015_valid_20141101093015_lead_024.nc
- init_20141031213015_valid_20141101213015_lead_024.nc
- init_20141101093015_valid_20141102093015_lead_024.nc

Forecast Lead: 120 hour

- init_20141026093015_valid_20141031093015_lead_120.nc

- init_20141026213015_valid_20141031213015_lead_120.nc
- init_20141027093015_valid_20141101093015_lead_120.nc
- init_20141031093015_valid_20141105093015_lead_120.nc
- init_20141031213015_valid_20141105213015_lead_120.nc
- init_20141101093015_valid_20141106093015_lead_120.nc

Keywords

Note:

- UserScriptUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.32.3 UserScript: Run Once Per Init Use Case

met_tool_wrapper/UserScript/UserScript_run_once_per_init.conf

Scientific Objective

Demonstrate how to run a user-defined script that should be executed once for each initialization time. This use case runs a simple ls command to list the contents of a directory. A wildcard character (*) is used to replace filename template tags for valid and lead to find all files that match any of the times available.

Datasets

Input: Empty test files from the METplus repository

METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command that is specified by the user.

METplus Workflow

UserScript is the only tool called in this example. It processes the following run times:

Init: 2014-10-31 09:30:15
Init: 2014-10-31 21:30:15
Init: 2014-11-01 09:30:15

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/UserScript/UserScript_run_once_per_init.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/UserScript/UserScript_
↪run_once_per_init.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H%M%S
INIT_BEG = 20141031093015
INIT_END = 20141101093015
INIT_INCREMENT = 12H

LEAD_SEQ = 0H, 12H, 24H, 120H

USER_SCRIPT_CUSTOM_LOOP_LIST = nc


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

USER_SCRIPT_INPUT_TEMPLATE = init_{init?fmt=%Y%m%d%H%M%S}_valid_{valid?fmt=%Y%m%d%H%M%S}_
↪lead_{lead?fmt=%3H}.{custom}
USER_SCRIPT_INPUT_DIR = {INPUT_BASE}/met_test/new/test



###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/met_tool_wrapper/UserScript/print_file_list.py
```

MET Configuration

None. UserScript does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in UserScript_run_once_per_init.conf_run_once then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/UserScript/
↪UserScript_run_once_per_init.conf_run_once -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_run_once_per_init.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/UserScript_
↪run_once_per_init.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

No output files are generated from this use case, but the logfile will contain the results of the directory listing command(s)

Init time: 2014-10-31 09:30:15

- init_20141031093015_valid_20141031093015_lead_000.nc
- init_20141031093015_valid_20141031213015_lead_012.nc
- init_20141031093015_valid_20141101093015_lead_024.nc
- init_20141031093015_valid_20141102093015_lead_048.nc
- init_20141031093015_valid_20141103093015_lead_072.nc
- init_20141031093015_valid_20141104093015_lead_096.nc
- init_20141031093015_valid_20141105093015_lead_120.nc
- init_20141031093015_valid_20141106093015_lead_144.nc
- init_20141031093015_valid_20141107093015_lead_168.nc

Init time: 2014-10-31 21:30:15

- init_20141031213015_valid_20141031213015_lead_000.nc
- init_20141031213015_valid_20141101093015_lead_012.nc
- init_20141031213015_valid_20141101213015_lead_024.nc
- init_20141031213015_valid_20141102213015_lead_048.nc
- init_20141031213015_valid_20141103213015_lead_072.nc
- init_20141031213015_valid_20141104213015_lead_096.nc
- init_20141031213015_valid_20141105213015_lead_120.nc
- init_20141031213015_valid_20141106213015_lead_144.nc
- init_20141031213015_valid_20141107213015_lead_168.nc

Init time: 2014-11-01 09:30:15

- init_20141101093015_valid_20141101093015_lead_000.nc
- init_20141101093015_valid_20141101213015_lead_012.nc
- init_20141101093015_valid_20141102093015_lead_024.nc
- init_20141101093015_valid_20141103093015_lead_048.nc
- init_20141101093015_valid_20141104093015_lead_072.nc
- init_20141101093015_valid_20141105093015_lead_096.nc
- init_20141101093015_valid_20141106093015_lead_120.nc
- init_20141101093015_valid_20141107093015_lead_144.nc
- init_20141101093015_valid_20141108093015_lead_168.nc

Keywords

Note:

- UserScriptUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.32.4 UserScript: Run Once Per Valid Use Case

met_tool_wrapper/UserScript/UserScript_run_once_per_valid.conf

Scientific Objective

Demonstrate how to run a user-defined script that should be executed once for each valid time. This use case runs a simple ls command to list the contents of a directory. A wildcard character (*) is used to replace filename template tags for init and lead to find all files that match any of the times available.

Datasets

Input: Empty test files from the METplus repository

METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command that is specified by the user.

METplus Workflow

UserScript is the only tool called in this example. It processes the following run times:

Valid: 2014-10-31 09:30:15
Valid: 2014-10-31 21:30:15
Valid: 2014-11-01 09:30:15

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/UserScript/UserScript_run_once_per_valid.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/UserScript/UserScript_
↪run_once_per_valid.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H%M%S
VALID_BEG = 20141031093015
VALID_END = 20141101093015
VALID_INCREMENT = 12H

LEAD_SEQ = 0H, 12H, 24H, 120H

USER_SCRIPT_CUSTOM_LOOP_LIST = nc
```

(continues on next page)

(continued from previous page)

```
###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

USER_SCRIPT_INPUT_TEMPLATE_LABELS = label0
USER_SCRIPT_INPUT_TEMPLATE = init_{init?fmt=%Y%m%d%H%M%S}_valid_{valid?fmt=%Y%m%d%H%M%S}_
↪lead_{lead?fmt=%3H}.{custom}
USER_SCRIPT_INPUT_DIR = {INPUT_BASE}/met_test/new/test


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/met_tool_wrapper/UserScript/print_file_list.py
```

MET Configuration

None. UserScript does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in UserScript_run_once_per_valid.conf_run_once then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/UserScript/
↪UserScript_run_once_per_valid.conf_run_once -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_run_once_per_valid.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/UserScript_
↪run_once_per_valid.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases

- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

No output files are generated from this use case, but the logfile will contain the results of the directory listing command(s)

Valid time: 2014-10-31 09:30:15

- init_20141024093015_valid_20141031093015_lead_168.nc
- init_20141025093015_valid_20141031093015_lead_144.nc
- init_20141026093015_valid_20141031093015_lead_120.nc
- init_20141027093015_valid_20141031093015_lead_096.nc
- init_20141028093015_valid_20141031093015_lead_072.nc
- init_20141029093015_valid_20141031093015_lead_048.nc
- init_20141030093015_valid_20141031093015_lead_024.nc
- init_20141030213015_valid_20141031093015_lead_012.nc
- init_20141031093015_valid_20141031093015_lead_000.nc

Valid time: 2014-10-31 21:30:15

- init_20141024213015_valid_20141031213015_lead_168.nc
- init_20141025213015_valid_20141031213015_lead_144.nc
- init_20141026213015_valid_20141031213015_lead_120.nc
- init_20141027213015_valid_20141031213015_lead_096.nc
- init_20141028213015_valid_20141031213015_lead_072.nc
- init_20141029213015_valid_20141031213015_lead_048.nc

- init_20141030213015_valid_20141031213015_lead_024.nc
- init_20141031093015_valid_20141031213015_lead_012.nc
- init_20141031213015_valid_20141031213015_lead_000.nc

Valid time: 2014-11-01 09:30:15

- init_20141025093015_valid_20141101093015_lead_168.nc
- init_20141026093015_valid_20141101093015_lead_144.nc
- init_20141027093015_valid_20141101093015_lead_120.nc
- init_20141028093015_valid_20141101093015_lead_096.nc
- init_20141029093015_valid_20141101093015_lead_072.nc
- init_20141030093015_valid_20141101093015_lead_048.nc
- init_20141031093015_valid_20141101093015_lead_024.nc
- init_20141031213015_valid_20141101093015_lead_012.nc
- init_20141101093015_valid_20141101093015_lead_000.nc

Keywords

Note:

- UserScriptUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.1.32.32.5 UserScript: Run Once Use Case

met_tool_wrapper/UserScript/UserScript_run_once.conf

Scientific Objective

Demonstrate how to run a user-defined script that should be executed one time. This use case runs a simple ls command to list the contents of a directory. A wildcard character (*) is used to replace filename template tags for init, valid, and lead to find all files that match any of the times available.

Datasets

Input: Empty test files from the METplus repository

METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command that is specified by the user.

METplus Workflow

UserScript is the only tool called in this example. It runs once with no time information specified.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/met_tool_wrapper/UserScript/UserScript_run_once.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/UserScript/UserScript_
↪run_once.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H%M%S
VALID_BEG = 20141031093015
VALID_END = 20141101093015
VALID_INCREMENT = 12H

LEAD_SEQ = 0H, 12H, 15H, 24H, 120H

USER_SCRIPT_CUSTOM_LOOP_LIST = nc


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

USER_SCRIPT_INPUT_TEMPLATE = init_{init?fmt=%Y%m%d%H%M%S}_valid_{valid?fmt=%Y%m%d%H%M%S}_
↪lead_{lead?fmt=%3H}.{custom}
USER_SCRIPT_INPUT_DIR = {INPUT_BASE}/met_test/new/test



###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/met_tool_wrapper/UserScript/print_file_list.py
```

MET Configuration

None. UserScript does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in UserScript_run_once.conf_run_once then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/UserScript/
↪UserScript_run_once.conf_run_once -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_run_once.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/met_tool_wrapper/GridStat/UserScript_
↪run_once.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

No output files are generated from this use case, but the logfile will contain the results of the directory listing command(s)

- init_20141031093015_valid_20141031213015_lead_012.nc

- init_20141031093015_valid_20141101093015_lead_024.nc
- init_20141031093015_valid_20141102093015_lead_048.nc
- init_20141031093015_valid_20141103093015_lead_072.nc
- init_20141031093015_valid_20141104093015_lead_096.nc
- init_20141031093015_valid_20141105093015_lead_120.nc
- init_20141031093015_valid_20141106093015_lead_144.nc
- init_20141031093015_valid_20141107093015_lead_168.nc
- init_20141031213015_valid_20141031213015_lead_000.nc
- init_20141031213015_valid_20141101093015_lead_012.nc
- init_20141031213015_valid_20141101213015_lead_024.nc
- init_20141031213015_valid_20141102213015_lead_048.nc
- init_20141031213015_valid_20141103213015_lead_072.nc
- init_20141031213015_valid_20141104213015_lead_096.nc
- init_20141031213015_valid_20141105213015_lead_120.nc
- init_20141031213015_valid_20141106213015_lead_144.nc
- init_20141031213015_valid_20141107213015_lead_168.nc
- init_20141101093015_valid_20141101093015_lead_000.nc
- init_20141101093015_valid_20141101213015_lead_012.nc
- init_20141101093015_valid_20141102093015_lead_024.nc
- init_20141101093015_valid_20141103093015_lead_048.nc
- init_20141101093015_valid_20141104093015_lead_072.nc
- init_20141101093015_valid_20141105093015_lead_096.nc
- init_20141101093015_valid_20141106093015_lead_120.nc
- init_20141101093015_valid_20141107093015_lead_144.nc
- init_20141101093015_valid_20141108093015_lead_168.nc

Keywords

Note:

- UserScriptUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.2 Model Applications

7.2.1 Air Quality and Composition

Data related to all areas of atmospheric composition, including ozone, smoke, dust, AOD and PM2.5

7.2.2 Climate

Average long range earth system predictions

7.2.3 Clouds

A category for use cases interested in the composition of, or characteristics associated with, clouds

7.2.4 Data Assimilation

Observational data used as part of the initial conditions for numerical weather prediction

7.2.5 Land Surface

Land Model diagnostics and verification against observations

7.2.6 Marine and Cryosphere

Data related to verification involving marine and cryosphere systems, which includes sea-ice

7.2.7 Medium Range

Lower resolution model configuration (>4km) usually producing forecasts out to 7-14 days (also referred to as global models)

7.2.8 Planetary Boundary Layer

Planetary Boundary Layer (PBL) applications

7.2.9 Precipitation

Any fields that can be defined as precipitation, including rain, snow, and other precipitation types

7.2.10 Subseasonal to Seasonal

Subseasonal-to-Seasonal model configurations; Lower resolution model configurations (>4km) usually producing forecasts out beyond 14 days and up 1 year

7.2.11 Subseasonal to Seasonal: Mid-Latitude

Subseasonal-to-Seasonal model configurations relating to middle latitudes

7.2.12 Subseasonal to Seasonal: Madden-Julian Oscillation

Subseasonal-to-Seasonal model configurations relating to the Madden-Julian oscillation

7.2.13 Short Range

High resolution model configurations (1-4km) usually producing forecasts between 0-3 days (also referred to as limited area models, stand-alone regional, and short range weather applications); Previously named Convection Allowing Models

7.2.14 Space Weather

Upper atmosphere and geospace model configurations

7.2.15 Tropical Cyclone and Extra Tropical Cyclone

Any field that is associated with Tropical Cyclone and Extra-tropical Cyclones

7.2.16 Unstructured Grids

Unstructured grids used by models for numerical weather prediction.

7.2.16.1 Air Quality and Composition

Data related to all areas of atmospheric composition, including ozone, smoke, dust, AOD and PM2.5

7.2.16.1.1 EnsembleStat: Using Python Embedding for Aerosol Optical Depth

model_applications/air_quality_and_comp/EnsembleStat_fcstICAP_obsMODIS_aod.conf

Scientific Objective

To provide useful statistical information on the relationship between observation data for aersol optical depth (AOD) to an ensemble forecast. These values can be used to help correct ensemble member deviations from observed values.

Datasets

Forecast: International Cooperative for Aerosol Prediction (ICAP) ensemble netCDF file, 7 members
Observation: Aggregate netCDF file with MODIS observed AOD field

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 788) section for more information.

METplus Components

This use case utilizes the METplus EnsembleStat wrapper to read in files using Python Embedding

METplus Workflow

EnsembleStat is the only tool called in this example. It processes a single run time with seven ensemble members. Three of the members do not have data for the AOD field, so EnsembleStat will only process four of the members for statistics.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/air_quality_and_comp/EnsembleStat_fcstICAP_obsMODIS_aod.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/air_quality_and_comp/
↪EnsembleStat_fcstICAP_obsMODIS_aod.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = EnsembleStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H%M
INIT_BEG=201608150000
INIT_END=201608150000
INIT_INCREMENT=06H
```

(continues on next page)

(continued from previous page)

```
LEAD_SEQ = 12H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_ENSEMBLE_STAT_INPUT_DATATYPE = PYTHON_NUMPY
FCST_ENSEMBLE_STAT_INPUT_DIR =
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE = 0, 1, 2, 3, 4, 5, 6

OBS_ENSEMBLE_STAT_INPUT_GRID_DATATYPE = PYTHON_NUMPY
OBS_ENSEMBLE_STAT_GRID_INPUT_DIR = {INPUT_BASE}/model_applications/air_quality_and_comp/aod
OBS_ENSEMBLE_STAT_GRID_INPUT_TEMPLATE = PYTHON_NUMPY

ENSEMBLE_STAT_OUTPUT_DIR = {OUTPUT_BASE}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/air_quality_and_comp/EnsembleStat_
↪fcstICAP_obsMODIS_aod

FCST_VAR1_NAME = {CONFIG_DIR}/forecast_embedded.py {OBS_ENSEMBLE_STAT_GRID_INPUT_DIR}/icap_
↪{init?fmt=%Y%m%d%H}_aod.nc:total_aod:{valid?fmt=%Y%m%d%H%M}:MET_PYTHON_INPUT_ARG

OBS_VAR1_NAME = {CONFIG_DIR}/analysis_embedded.py {OBS_ENSEMBLE_STAT_GRID_INPUT_DIR}/AGGR_
↪HOURLY_{valid?fmt=%Y%m%d}T{valid?fmt=%H%M}_1deg_global_archive.nc:aod_nrl_total:Mean


###
# EnsembleStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ensemblestat
###

MODEL = ICAP
OBTYPE = NRL_AOD

ENSEMBLE_STAT_OBS_WINDOW_BEGIN = -5400
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_OBS_WINDOW_END = 5400

ENSEMBLE_STAT_N_MEMBERS = 7

ENSEMBLE_STAT_ENS_THRESH = 0.1

ENSEMBLE_STAT_REGRID_TO_GRID = NONE

ENSEMBLE_STAT_OUTPUT_PREFIX =
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *EnsembleStat MET Configuration* (page 105) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Ensemble-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
${METPLUS_DESC}

//
```

(continues on next page)

(continued from previous page)

```
// Output observation type to be written
//
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
${METPLUS_CENSOR_THRESH}
${METPLUS_CENSOR_VAL}
cat_thresh    = [];
nc_var_str    = "";

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}


////////////////////////////////////////////////////////////////////////////////

//prob_cat_thresh =
${METPLUS_PROB_CAT_THRESH}

//prob_pct_thresh =
${METPLUS_PROB_PCT_THRESH}

//eclv_points =
${METPLUS_ECLV_POINTS}


////////////////////////////////////////////////////////////////////////////////

//
// Forecast and observation fields to be verified
//
```

(continues on next page)

(continued from previous page)

```
fcst = {

   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_ENS_THRESH}
   ${METPLUS_VLD_THRESH}
   ${METPLUS_FCST_FIELD}
}

obs = {

   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//

${METPLUS_MESSAGE_TYPE}
sid_exc        = [];
//obs_thresh     = [ NA ];
${METPLUS_OBS_THRESH}

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

${METPLUS_DUPLICATE_FLAG}
obs_summary    = NONE;
obs_perc_value = 50;
${METPLUS_SKIP_CONST}

//
// Observation error options
// Set dist_type to NONE to use the observation error table instead
// May be set separately in each "obs.field" entry
//
obs_error = {
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_OBS_ERROR_FLAG}
   dist_type        = NONE;
   dist_parm        = [];
   inst_bias_scale  = 1.0;
   inst_bias_offset = 0.0;
   min              = NA;      // Valid range of data
   max              = NA;
}

//
// Mapping of message type group name to comma-separated list of values.
//
message_type_group_map = [
   { key = "SURFACE"; val = "ADPSFC,SFCSHP,MSONET";               },
   { key = "ANYAIR";  val = "AIRCAR,AIRCFT";                      },
   { key = "ANYSFC";  val = "ADPSFC,SFCSHP,ADPUPA,PROFLR,MSONET"; },
   { key = "ONLYSF";  val = "ADPSFC,SFCSHP";                      }
];

//
// Ensemble bin sizes
// May be set separately in each "obs.field" entry
//
${METPLUS_ENS_SSVAR_BIN_SIZE}
${METPLUS_ENS_PHIST_BIN_SIZE}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}



//
// May be set separately in each "obs.field" entry
//
${METPLUS_CLIMO_CDF_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
mask = {
   ${METPLUS_MASK_GRID}
   ${METPLUS_MASK_POLY}
   sid    = [];
   llpnt = [];
}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
${METPLUS_CI_ALPHA}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gridded verification output types
// May be set separately in each "obs.field" entry
```

(continues on next page)

(continued from previous page)

```
//
${METPLUS_NC_ORANK_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Random number generator
//
rng = {
   type = "mt19937";
   seed = "1";
}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

${METPLUS_OUTPUT_PREFIX}
//version          = "V9.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses two Python embedding scripts to read input data

parm/use_cases/model_applications/air_quality_and_comp/EnsembleStat_fcstICAP_obsMODIS_aod/forecast_embedd

```
import sys
import re
import numpy as np
import datetime as dt
from netCDF4 import Dataset, chartostring

#grab input from user
#should be (1)input file using full path (2) variable name (3) valid time for the forecast␣
↪in %Y%m%d%H%M format and (4) ensemble member number, all separated by ':' characters
#program can only accept that 1 input, while still maintaining user flexability to change␣
↪multiple
#variables, including valid time, ens member, etc.
```

(continues on next page)

(continued from previous page)

```
input_file, var_name, val_time, ens_mem = sys.argv[1].split(':')
ens_mem = int(ens_mem)
val_time = dt.datetime.strptime(val_time,"%Y%m%d%H%M")
try:
    #set pointers to file and group name in file
    f = Dataset(input_file, 'r')
    v = f[var_name]
    #grab intialization time from file name and hold
    #also compute the lead time
    i_time_ind = input_file.split("_").index("aod.nc")-1
    i_time = input_file.split("_")[i_time_ind]
    i_time_obj = dt.datetime.strptime(i_time,"%Y%m%d%H")
    lead, rem = divmod((val_time - i_time_obj).total_seconds(), 3600)

    print("Ensemble Member evaluation for: "+f.members.split(',')[ens_mem])

    #checks if the the valid time for the forecast from user is present in file.
    #Exits if the time is not present with a message
    if not val_time.timestamp() in f['time'][:]:
            print("valid time of "+str(val_time)+" is not present. Check file initialization␣
↪time, passed valid time.")
            f.close()
            sys.exit(1)

    #grab index in the time array for the valid time provided by user (val_time)
    val_time_ind = np.where(f['time'][:] == val_time.timestamp())[0][0]

    #grab data from file
    lat = np.float64(f.variables['lat'][:])
    lon = np.float64(f.variables['lon'][:])
    var = np.float64(v[val_time_ind:val_time_ind+1,ens_mem:ens_mem+1,::-1,:])
    var[var < -800] = -9999
    #squeeze out all 1d arrays, add fill value
    met_data = np.squeeze(var).copy()
except NameError:
    print("Can't find input file")
    sys.exit(1)

##########
#create a metadata dictionary

attrs = {

        'valid': str(val_time.strftime("%Y%m%d"))+'_'+str(val_time.strftime("%H%M%S")),
        'init': i_time[:-2]+'_'+i_time[-2:]+'0000',
```

(continues on next page)

(continued from previous page)

```
        'name': var_name,
        'long_name': 'UNKNOWN',
        'lead': str(int(lead)),
        'accum': '00',
        'level': 'UNKNOWN',
        'units': 'UNKNOWN',

        'grid': {
            'name': 'Global 1 degree',
            'type': 'LatLon',
            'lat_ll': -89.5,
            'lon_ll': -179.5,
            'delta_lat': 1.0,
            'delta_lon': 1.0,

            'Nlon': f.dimensions['lon'].size,
            'Nlat': f.dimensions['lat'].size,
            }
        }

#print some output to show script ran successfully
print("Input file: " + repr(input_file))
print("Variable name: " + repr(var_name))
print("valid time: " + repr(val_time.strftime("%Y%m%d%H%M")))
print("Attributes:\t" + repr(attrs))
f.close()
```

parm/use_cases/model_applications/air_quality_and_comp/EnsembleStat_fcstICAP_obsMODIS_aod/analysis_embedd

```
import sys
import re
import numpy as np
import datetime as dt
from netCDF4 import Dataset, chartostring

#grab input from user
#should be (1)input file using full path (2) group name for the variable and (3) variable␣
↪name
input_file, group_name, var_name = sys.argv[1].split(':')
try:
    #set pointers to file and group name in file
    f = Dataset(input_file, 'r')
    g = f.groups[group_name]
    #grab time from file name and hold
    v_time_ind = input_file.split("_").index("HOURLY")+1
    v_time = input_file.split("_")[v_time_ind]
```

(continues on next page)

(continued from previous page)

```
    #grab data from file
    lat = np.float64(f.variables['latitude'][:])
    lon = np.float64(f.variables['longitude'][:])
    #the data is defined by (lon, lat), so it needs to be transposed
    #in addition to being filled by fill value if data is missing
    var_invert = np.float64(g.variables[var_name][:,::-1])
    var_invert[var_invert < -800] = -9999
    met_data = var_invert.T.copy()
except NameError:
    print("Can't find input file")
    sys.exit(1)

##########

#create a metadata dictionary

attrs = {

        'valid': str(v_time.split('T')[0])+'_'+str(v_time.split('T')[1])+'00',
        'init': str(v_time.split('T')[0])+'_'+str(v_time.split('T')[1])+'00',
        'name': group_name+'_'+var_name,
        'long_name': 'UNKNOWN',
        'lead': '00',
        'accum': '00',
        'level': 'UNKNOWN',
        'units': 'UNKNOWN',

        'grid': {
            'name': 'Global 1 degree',
            'type': 'LatLon',
            'lat_ll': -89.5,
            'lon_ll': -179.5,
            'delta_lat': 1.0,
            'delta_lon': 1.0,

            'Nlon': f.dimensions['longitude'].size,
            'Nlat': f.dimensions['latitude'].size,
            }
        }

#print some output to show script ran successfully
print("Input file: " + repr(input_file))
print("Group name: " + repr(group_name))
print("Variable name: " + repr(var_name))
```

(continues on next page)

(continued from previous page)

```
print("Attributes:\t" + repr(attrs))
f.close()
```

Running METplus

It is recommended to run this use case by:

Passing in EnsembleStat_python_embedding.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/air_quality_and_comp/
↪EnsembleStat_fcstICAP_obsMODIS_aod.conf -c /path/to/user_system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/air_quality/AOD (relative to **OUTPUT_BASE**) and will contain the following files:

- ensemble_stat_aod_20160815_120000V_ecnt.txt
- ensemble_stat_aod_20160815_120000V_ens.nc
- ensemble_stat_aod_20160815_120000V_orank.nc
- ensemble_stat_aod_20160815_120000V_phist.txt
- ensemble_stat_aod_20160815_120000V_relp.txt

- ensemble_stat_aod_20160815_120000V_rhist.txt
- ensemble_stat_aod_20160815_120000V_ssvar.txt
- ensemble_stat_aod_20160815_120000V.stat

Keywords

Note:

- EnsembleStatToolUseCase
- PythonEmbeddingFileUseCase
- AirQualityAndCompAppUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/air_quality_and_comp-EnsembleStat_fcstICAP_obsMODIS_aod.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.2 Climate

Average long range earth system predictions

7.2.16.2.1 MODE: CESM and GPCP Asian Monsoon Precipitation

model_applications/climate/MODE_fcstCESM_obsGPCP_AsianMonsoonPrecip.conf

Scientific Objective

To evaluate the CESM model daily precipitation against the GPCP daily precipitation over the Indian Monsoon region to obtain object based output statistics. This was developed as part of the NCAR System for Integrated Modeling of the Atmosphere (SIMA) project.

Datasets

- Forecast dataset: CESM Daily Precipitation
- Observation dataset: GPCP Daily Precipitation

METplus Components

This use case runs mode to create object based statistics on daily precipitation data from the CESM model and observations from the GPCP.

METplus Workflow

The mode tool is run for each time. This example loops by model initialization time. It processes 4 valid times, listed below.

Valid: 2014-08-02
Forecast lead: 24

Init: 2014-08-03
Forecast lead: 48

Init: 2014-08-03
Forecast lead: 24

Init: 2014-08-04
Forecast lead: 48

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/climate/MODE_fcstCESM_obsGPCP_AsianMonsoonPrecip.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/climate/MODE_
↪fcstCESM_obsGPCP_AsianMonsoonPrecip.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = Mode


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2014060100
INIT_END = 2014060200
INIT_INCREMENT = 86400

LEAD_SEQ = 24, 48


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

FCST_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/climate/CESM
FCST_MODE_INPUT_TEMPLATE = MetPlus.globe.{init?fmt=%Y-%m-%d}-00000.cam.h1.{init?fmt=%Y-%m-%d?
↪shift=86400}-00000.nc

OBS_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/climate/GPCP
OBS_MODE_INPUT_TEMPLATE = gpcp_v01r03_daily_d{valid?fmt=%Y%m%d?shift=-86400}_c20170530.nc

MODE_OUTPUT_DIR = {OUTPUT_BASE}/climate/CESM_MODE
MODE_OUTPUT_TEMPLATE = {init?fmt=%Y_%m_%d_%H%M%S}

MODE_VERIFICATION_MASK_TEMPLATE = {FCST_MODE_INPUT_DIR}/asia_monsoon_cesm_mask.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = CESM
FCST_VAR1_NAME = PRECT
FCST_VAR1_LEVELS = "({valid?fmt=%Y%m%d_%H%M%S},*,*)"
FCST_VAR1_OPTIONS = convert(x) = 86400000*x;

MODE_FCST_FILTER_ATTR_NAME = AREA
MODE_FCST_FILTER_ATTR_THRESH = >=7

MODE_OBS_FILTER_ATTR_NAME = AREA
MODE_OBS_FILTER_ATTR_THRESH = >=7


OBTYPE = GPCP
OBS_VAR1_NAME = precip
OBS_VAR1_LEVELS = "(0,*,*)"


###
# MODE Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mode
###

MODE_GRID_RES = 1
```

(continues on next page)

(continued from previous page)

```
MODE_QUILT = True

MODE_CONV_RADIUS = 2

MODE_CONV_THRESH = ge12.0, ge25.0

MODE_MERGE_THRESH = ge10.0, ge20.0

MODE_MERGE_FLAG = THRESH

MODE_MATCH_FLAG = NO_MERGE

MODE_NC_PAIRS_FLAG_POLYLINES = False

MODE_MASK_POLY_FLAG = BOTH

MODE_WEIGHT_ASPECT_DIFF = 1.0

MODE_REGRID_TO_GRID = FCST

MODE_MASK_MISSING_FLAG = BOTH
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *MODE MET Configuration* (page 174) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Approximate grid resolution (km)
//
// grid_res =
${METPLUS_GRID_RES}

////////////////////////////////////////////////////////////////////////////////

//
// Run all permutations of radius and threshold
//
// quilt =
${METPLUS_QUILT}

//
// MODE Multivar boolean combination logic
//
```

(continues on next page)

(continued from previous page)

```
//multivar_logic =
${METPLUS_MULTIVAR_LOGIC}

//multivar_intensity_flag =
${METPLUS_MULTIVAR_INTENSITY_FLAG}

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FIELD}

   ${METPLUS_FCST_CENSOR_THRESH}
   ${METPLUS_FCST_CENSOR_VAL}
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}
   ${METPLUS_FCST_VLD_THRESH}
   ${METPLUS_FCST_FILTER_ATTR_NAME}
   ${METPLUS_FCST_FILTER_ATTR_THRESH}
   ${METPLUS_FCST_MERGE_THRESH}
   ${METPLUS_FCST_MERGE_FLAG}
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_MULTIVAR_NAME}
   ${METPLUS_FCST_MULTIVAR_LEVEL}
}

obs = {
   ${METPLUS_OBS_FIELD}

   ${METPLUS_OBS_CENSOR_THRESH}
   ${METPLUS_OBS_CENSOR_VAL}
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
   ${METPLUS_OBS_VLD_THRESH}
   ${METPLUS_OBS_FILTER_ATTR_NAME}
   ${METPLUS_OBS_FILTER_ATTR_THRESH}
   ${METPLUS_OBS_MERGE_THRESH}
   ${METPLUS_OBS_MERGE_FLAG}
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_MULTIVAR_NAME}
   ${METPLUS_OBS_MULTIVAR_LEVEL}
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Handle missing data
//
// mask_missing_flag =
${METPLUS_MASK_MISSING_FLAG}

//
// Match objects between the forecast and observation fields
//
//match_flag =
${METPLUS_MATCH_FLAG}

//
// Maximum centroid distance for objects to be compared
//
//max_centroid_dist =
${METPLUS_MAX_CENTROID_DIST}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine weights
//
//weight = {
${METPLUS_WEIGHT_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine interest functions
//
interest_function = {

   ${METPLUS_INTEREST_FUNCTION_CENTROID_DIST}

   ${METPLUS_INTEREST_FUNCTION_BOUNDARY_DIST}
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_INTEREST_FUNCTION_CONVEX_HULL_DIST}

   angle_diff = (
      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 0.0 )
   );

   aspect_diff = (
      (  0.00, 1.0 )
      (  0.10, 1.0 )
      (  0.75, 0.0 )
   );

   corner   = 0.8;
   ratio_if = (
      (    0.0, 0.0 )
      ( corner, 1.0 )
      (    1.0, 1.0 )
   );

   area_ratio = ratio_if;

   int_area_ratio = (
      ( 0.00, 0.00 )
      ( 0.10, 0.50 )
      ( 0.25, 1.00 )
      ( 1.00, 1.00 )
   );

   curvature_ratio = ratio_if;

   complexity_ratio = ratio_if;

   inten_perc_ratio = ratio_if;
}

////////////////////////////////////////////////////////////////////////////////

//
// Total interest threshold for determining matches
//
//total_interest_thresh =
${METPLUS_TOTAL_INTEREST_THRESH}
```

(continues on next page)

(continued from previous page)

```
//
// Interest threshold for printing output pair information
//
print_interest_thresh = 0.0;

////////////////////////////////////////////////////////////////////////////////

//
// Plotting information
//
met_data_dir = "MET_BASE";

fcst_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

obs_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

object_plot = {
   color_table      = "MET_BASE/colortables/mode_obj.ctable";
}

//
// Boolean for plotting on the region of valid data within the domain
//
plot_valid_flag = FALSE;

//
// Plot polyline edges using great circle arcs instead of straight lines
//
plot_gcarc_flag = FALSE;

////////////////////////////////////////////////////////////////////////////////

//
// NetCDF matched pairs, PostScript, and contingency table output files
//
//ps_plot_flag =
${METPLUS_PS_PLOT_FLAG}
```

(continues on next page)

(continued from previous page)

```
//nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

//ct_stats_flag =
${METPLUS_CT_STATS_FLAG}


////////////////////////////////////////////////////////////////////////////////

shift_right = 0;   //  grid squares

////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in MODE_fcstCESM_obsGPCP_ConusPrecip.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/climate/MODE_
↪fcstCESM_obsGPCP_AsianMonsoonPrecip.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in MODE_fcstCESM_obsGPCP_AsianMonsoonPrecip.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/climate/MODE_
↪fcstCESM_obsGPCP_AsianMonsoonPrecip.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/climate/CESM_MODE (relative to **OUTPUT_BASE**) and will contain the following files:

2014_06_01_000000/mode_000000L_20140602_000000V_000000A_R1_T1_cts.txt
2014_06_01_000000/mode_000000L_20140602_000000V_000000A_R1_T1_obj.nc
2014_06_01_000000/mode_000000L_20140602_000000V_000000A_R1_T1_obj.txt
2014_06_01_000000/mode_000000L_20140602_000000V_000000A_R1_T1.ps
2014_06_01_000000/mode_000000L_20140602_000000V_000000A_R1_T2_cts.txt
2014_06_01_000000/mode_000000L_20140602_000000V_000000A_R1_T2_obj.nc
2014_06_01_000000/mode_000000L_20140602_000000V_000000A_R1_T2_obj.txt
2014_06_01_000000/mode_000000L_20140602_000000V_000000A_R1_T2.ps
2014_06_01_000000/mode_000000L_20140603_000000V_000000A_R1_T1_cts.txt
2014_06_01_000000/mode_000000L_20140603_000000V_000000A_R1_T1_obj.nc
2014_06_01_000000/mode_000000L_20140603_000000V_000000A_R1_T1_obj.txt
2014_06_01_000000/mode_000000L_20140603_000000V_000000A_R1_T1.ps
2014_06_01_000000/mode_000000L_20140603_000000V_000000A_R1_T2_cts.txt
2014_06_01_000000/mode_000000L_20140603_000000V_000000A_R1_T2_obj.nc
2014_06_01_000000/mode_000000L_20140603_000000V_000000A_R1_T2_obj.txt
2014_06_01_000000/mode_000000L_20140603_000000V_000000A_R1_T2.ps
2014_06_02_000000/mode_000000L_20140603_000000V_000000A_R1_T1_cts.txt
2014_06_02_000000/mode_000000L_20140603_000000V_000000A_R1_T1_obj.nc
2014_06_02_000000/mode_000000L_20140603_000000V_000000A_R1_T1_obj.txt
2014_06_02_000000/mode_000000L_20140603_000000V_000000A_R1_T1.ps
2014_06_02_000000/mode_000000L_20140603_000000V_000000A_R1_T2_cts.txt
2014_06_02_000000/mode_000000L_20140603_000000V_000000A_R1_T2_obj.nc
2014_06_02_000000/mode_000000L_20140603_000000V_000000A_R1_T2_obj.txt
2014_06_02_000000/mode_000000L_20140603_000000V_000000A_R1_T2.ps
2014_06_02_000000/mode_000000L_20140604_000000V_000000A_R1_T1_cts.txt
2014_06_02_000000/mode_000000L_20140604_000000V_000000A_R1_T1_obj.nc
2014_06_02_000000/mode_000000L_20140604_000000V_000000A_R1_T1_obj.txt

2014_06_02_000000/mode_000000L_20140604_000000V_000000A_R1_T1.ps
2014_06_02_000000/mode_000000L_20140604_000000V_000000A_R1_T2_cts.txt
2014_06_02_000000/mode_000000L_20140604_000000V_000000A_R1_T2_obj.nc
2014_06_02_000000/mode_000000L_20140604_000000V_000000A_R1_T2_obj.txt
2014_06_02_000000/mode_000000L_20140604_000000V_000000A_R1_T2.ps

Keywords

Note:

- MODEToolUseCase
- ClimateAppUseCase
- NetCDFFileUseCase
- NCAROrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/climate-MODE_fcstCESM_obsGPCP_AsianMonsoonPrecip.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.2.2 Grid-Stat: CESM and GFS Analysis CONUS Temp

model_applications/climate/ GridStat_fcstCESM_obsGFS _ConusTemp.conf

Scientific Objective

To evaluate the CESM model temperature against the GFS analysis across the the Continental United States to obtain categorical output statistics. This was developed as part of the NCAR System for Integrated Modeling of the Atmosphere (SIMA) project.

Datasets

- Forecast dataset: CESM Surface Temperature Data
- Observation dataset: GFS Analysis 2m Temperature

METplus Components

This use case runs grid_stat to create continuous statistics on tempeprature from the CESM model and observations from the GFS analysis.

METplus Workflow

The grid_stat tool is run for each time. This example loops by initialization time. It processes 4 valid times, listed below.

Valid: 2014-08-01_06Z
Forecast lead: 06

Init: 2014-08-01_12Z
Forecast lead: 12

Init: 2014-08-02_06Z
Forecast lead: 06

Init: 2014-08-02_12Z
Forecast lead: 12

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/climate/GridStat_fcstCESM_obsGFS_ConusTemp.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/climate/GridStat_
↪fcstCESM_obsGFS_ConusTemp.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###
```

(continues on next page)

(continued from previous page)

```
PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2014080100
INIT_END = 2014080200
INIT_INCREMENT = 86400

LEAD_SEQ = 6, 12


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/climate/CESM
FCST_GRID_STAT_INPUT_TEMPLATE = MetPlus.globe.{init?fmt=%Y-%m-%d}-00000.cam.h0.{init?fmt=%Y-
↪%m-%d}-10800.nc

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/climate/gfs_analysis
OBS_GRID_STAT_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/gfsanl_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H
↪%M}_000.grb2

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/climate/CESM_GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}


###
# Field Info
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

GRID_STAT_ONCE_PER_FIELD = False

FCST_VAR1_NAME = TS
FCST_VAR1_LEVELS = "({valid?fmt=%Y%m%d_%H%M%S},*,*)"
FCST_VAR1_THRESH = ge32.0, ge65.0, ge75.0
FCST_VAR1_OPTIONS = convert(x) = K_to_F(x);

OBS_VAR1_NAME = TMP
OBS_VAR1_LEVELS = Z2
OBS_VAR1_THRESH = ge32.0, ge65.0, ge75.0
OBS_VAR1_OPTIONS = convert(x) = K_to_F(x);


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

MODEL = CESM
OBTYPE = GFS_ANALYS

GRID_STAT_REGRID_TO_GRID = FCST

GRID_STAT_VERIFICATION_MASK = {FCST_GRID_STAT_INPUT_DIR}/conus_cesm_mask.nc

GRID_STAT_OUTPUT_PREFIX={MODEL}_{CURRENT_OBS_NAME}_vs_{OBTYPE}

GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_CNT = STAT
GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE

GRID_STAT_INTERP_FIELD = NONE
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
```

(continues on next page)

(continued from previous page)

```
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
```

(continues on next page)

(continued from previous page)

```
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstCESM_obsGFS_ConusTemp.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/climate/GridStat_
↪fcstCESM_obsGFS_ConusTemp.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstCESM_obsGFS_ConusTemp.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/climate/GridStat_
↪fcstCESM_obsGFS_ConusTemp.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/climate/CESM_GridStat/grid_stat (relative to **OUTPUT_BASE**) and will contain the following files:

grid_stat_CESM_TMP_vs_GFS_ANALYS_060000L_20140801_060000V.stat grid_stat_CESM_TMP_vs_GFS_ANALYS_12
grid_stat_CESM_TMP_vs_GFS_ANALYS_060000L_20140802_060000V.stat grid_stat_CESM_TMP_vs_GFS_ANALYS_12

Keywords

Note:

- GridStatToolUseCase
- ClimateAppUseCase
- NetCDFFileUseCase
- NCAROrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/climate-GridStat_fcstCESM_obsGFS_ConusTemp.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.3 Clouds

A category for use cases interested in the composition of, or characteristics associated with, clouds

7.2.16.3.1 GridStat: Cloud Fractions with Neighborhood and Probabilities

model_applications/clouds/GridStat_fcstGFS_obsERA5_lowAndTotalCloudFrac.conf

Scientific Objective

This use case captures various statistical measures of two model comparisons for low and total cloud fractions with different neighborhood settings for internal model metrics and to aid in future model updates

Datasets

Forecast: Global Forecast System (GFS)
Observations: ECMWF Reanalysis, Version 5 (ERA5)
Grid: GPP 17km masking region

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

METplus Components

This use case utilizes Python Embedding, which is called using the PYTHON_NUMPY keyword in the observation input template settings. The same Python script can processes both forecast and observation datasets, but only the observation dataset is not set up for native ingest by MET. Two separate forecast fields are verified against two respective observation fields, with the Python script being passed the input file, the model name, the variable name being analyzed, the initialization and valid times, and a flag to indicate if the field passed is observation or forecast. This process is repeated with 3 instance names to GridStat, each with a different setting for regridding, neighborhood evaluation, thresholding, output line types, and output prefix names.

METplus Workflow

GridStat is the only MET tool called in this example. It processes the following run time:

Init: 2022-07-03 12Z
Forecast lead: 36 hour

Because instance names are used, GridStat will run 3 times for this 1 initalization time.

METplus Configuration

METplus first loads the default configuration file found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line: parm/use_cases/model_applications/clouds/GridStat_fcstGFS_obsERA5_lowAndTotalCloudFrac.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/clouds/GridStat_
↪fcstGFS_obsERA5_lowAndTotalCloudFrac.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

# ###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat, GridStat(nbr), GridStat(prob)

###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
```

(continues on next page)

(continued from previous page)

```
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2022070312
INIT_END=2022070312
INIT_INCREMENT = 12H

LEAD_SEQ = 36

LOOP_ORDER = times

###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/clouds/GridStat_fcstGFS_obsERA5_
↪lowAndTotalCloudFrac
FCST_GRID_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d}_gfs.t12z.pgrb2.0p25.f0{LEAD_SEQ}

OBS_GRID_STAT_INPUT_DIR =
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_CLIMO_MEAN_INPUT_DIR =
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/clouds/GridStat_fcstGFS_obsERA5_
↪lowAndTotalCloudFrac
GRID_STAT_OUTPUT_TEMPLATE =


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = GFS
OBTYPE = ERA5

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/clouds/GridStat_fcstGFS_obsERA5_
↪lowAndTotalCloudFrac

FCST_VAR1_NAME = TCDC
```

(continues on next page)

(continued from previous page)

```
FCST_VAR1_LEVELS = R636
FCST_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0

OBS_VAR1_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstGFS_obsERA5_lowAndTotalCloudFrac/ERA5_{valid?fmt=%Y%m%d}00_Cld.nc:{OBTYPE}
↪:totalCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:2
OBS_VAR1_LEVELS =
OBS_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0


FCST_VAR2_NAME = LCDC
FCST_VAR2_LEVELS = R630
FCST_VAR2_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0

OBS_VAR2_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstGFS_obsERA5_lowAndTotalCloudFrac/ERA5_{valid?fmt=%Y%m%d}00_Cld.nc:{OBTYPE}
↪:lowCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:2
OBS_VAR2_THRESH =  gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

#LOG_GRID_STAT_VERBOSITY = 2

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

GRID_STAT_REGRID_TO_GRID = FCST
GRID_STAT_REGRID_METHOD = BILIN
GRID_STAT_REGRID_WIDTH = 2

GRID_STAT_DESC =

FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0
OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_ONCE_PER_FIELD = False

FCST_IS_PROB = false

FCST_GRID_STAT_PROB_THRESH = ==0.1

OBS_IS_PROB = false

OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudFracs

GRID_STAT_OUTPUT_FLAG_FHO = STAT
GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_CNT = STAT
GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_MASK_POLY = {INPUT_BASE}/model_applications/clouds/GridStat_fcstGFS_obsERA5_
↪lowAndTotalCloudFrac/GPP_17km_60S_60N_mask.nc

[nbr]

FCST_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SFP20, >SFP30, >SFP40, >SFP50, >SFP60, >SFP70, >SFP80
OBS_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SOP20, >SOP30, >SOP40, >SOP50, >SOP60, >SOP70, >SOP80

FCST_VAR2_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SFP20, >SFP30, >SFP40, >SFP50, >SFP60, >SFP70, >SFP80
OBS_VAR2_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SOP20, >SOP30, >SOP40, >SOP50, >SOP60, >SOP70, >SOP80
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9
GRID_STAT_NEIGHBORHOOD_SHAPE = CIRCLE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >0.0

GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = NONE
GRID_STAT_OUTPUT_FLAG_CTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = NONE
GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTC = STAT
GRID_STAT_OUTPUT_FLAG_NBRCTS = STAT
GRID_STAT_OUTPUT_FLAG_NBRCNT = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = NONE

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_NBRHD = TRUE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudFracs_NBR

[prob]

FCST_IS_PROB = TRUE

FCST_VAR1_THRESH = >0.1, >0.2, >0.3, >0.4, >0.5, >0.6, >0.7, >0.8, >0.9, >1.0
OBS_VAR1_THRESH = gt0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.0, ge90.0

FCST_VAR2_THRESH = >0.1, >0.2, >0.3, >0.4, >0.5, >0.6, >0.7, >0.8, >0.9, >1.0
OBS_VAR2_THRESH = gt0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.0, ge90.0

GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9
GRID_STAT_NEIGHBORHOOD_SHAPE = CIRCLE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >0.0

GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = NONE
GRID_STAT_OUTPUT_FLAG_CTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = NONE
GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_OUTPUT_FLAG_NBRCTC = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTS = NONE
GRID_STAT_OUTPUT_FLAG_NBRCNT = NONE
GRID_STAT_OUTPUT_FLAG_GRAD = NONE
GRID_STAT_OUTPUT_FLAG_PCT = STAT
GRID_STAT_OUTPUT_FLAG_PSTD = STAT
GRID_STAT_OUTPUT_FLAG_PJC = STAT
GRID_STAT_OUTPUT_FLAG_PRC = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_NBRHD = FALSE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudFracs_PROB
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
```

(continues on next page)

(continued from previous page)

```
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
```

(continues on next page)

(continued from previous page)

```
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
```

(continues on next page)

(continued from previous page)

```
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case utilizes 1 Python script to read and process the observation fields. parm/use_cases/model_applications/clouds/GridStat_fcstGFS_obsERA5_lowAndTotalCloudFrac/read_input_data.py

```
#this code was provided by Craig Schwartz
#and is largely unaltered from its original
#function.

#from __future__ import print_function
import os
import sys
import numpy as np
import datetime as dt
from netCDF4 import Dataset  # http://code.google.com/p/netcdf4-python/
from scipy.interpolate import NearestNDInterpolator, LinearNDInterpolator
#### for Plotting
import matplotlib.cm as cm
import matplotlib.axes as maxes
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
#from mpl_toolkits.basemap import Basemap
import fnmatch
import pygrib
import pickle as pk
#####

##########################################

missing_values = -9999.0  # for MET

# UPP top layer bounds (Pa) for cloud layers
PTOP_LOW_UPP  = 64200. # low for > 64200 Pa
PTOP_MID_UPP  = 35000. # mid between 35000-64200 Pa
PTOP_HIGH_UPP = 15000. # high between 15000-35000 Pa

# Values for 4 x 4 contingency table
Na, Nb, Nc, Nd = 1, 2, 3, 4
Ne, Nf, Ng, Nh = 5, 6, 7, 8
Ni, Nj, Nk, Nl = 9, 10, 11, 12
```

(continues on next page)

(continued from previous page)

```
Nm, Nn, No, Np = 13, 14, 15, 16

# Notes:
# 1) Entry for 'point' is for point-to-point comparison and is all dummy data (except for␣
↪gridType) that is overwritten by point2point
# 2) ERA5 on NCAR CISL RDA changed at some point.  Old is ERA5_2017 (not used anymore), new␣
↪is ERA5, which we'll use for 2020 data
griddedDatasets =  {
   'MERRA2'    : { 'gridType':'LatLon', 'latVar':'lat',      'latDef':[-90.0,0.50,361], 'lonVar
↪':'lon',        'lonDef':[-180.0,0.625,576],   'flipY':True, 'ftype':'nc'},
   'SATCORPS' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[-180.0,0.3125,1152], 'flipY':False, 'ftype':'nc' },
   'ERA5_2017': { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.7848769072,0.
↪281016829130516,640], 'lonVar':'longitude', 'lonDef':[0.0,0.28125,1280], 'flipY':False,
↪'ftype':'nc' },
   'ERA5'      : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'GFS'       : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'GALWEM'    : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':True, 'ftype':'grib'},
   'GALWEM17' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.921875,0.156250,
↪1152], 'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536], 'flipY':False, 'ftype':
↪'grib'},
   'WWMCA'     : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'MPAS'      : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721],
↪'lonVar':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'nc'},
   'SAT_WWMCA_MEAN' : { 'gridType':'LatLon', 'latVar':'lat','latDef':[-90.0,0.25,721],
↪'lonVar':'lon', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'point'     : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.156250,1152],
↪'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536],   'flipY':False, 'ftype':'nc'},
}
   #TODO:Correct one, but MET can ingest a Gaussian grid only in Grib2 format (from Randy B.)
   #'ERA5'      : { 'gridType':'Gaussian', 'nx':1280, 'ny':640, 'lon_zero':0, 'latVar':
↪'latitude', 'lonVar':'longitude', 'flipY':False, },

#GALWEM, both 17-km and 0.25-degree
lowCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':3,
↪'typeOfFirstFixedSurface':10, 'shortName':'lcc' }
midCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':4,
↪'typeOfFirstFixedSurface':10, 'shortName':'mcc' }
highCloudFrac_GALWEM =  { 'parameterCategory':6, 'parameterNumber':5,
↪'typeOfFirstFixedSurface':10, 'shortName':'hcc' }
totalCloudFrac_GALWEM = { 'parameterCategory':6, 'parameterNumber':1,
```

(continues on next page)

(continued from previous page)

```
↪'typeOfFirstFixedSurface':10, 'shortName':'tcc' }
cloudTopHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':12,
↪'typeOfFirstFixedSurface':3, 'shortName':'cdct' }
cloudBaseHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':11,
↪'typeOfFirstFixedSurface':2, 'shortName':'cdcb' }

#GFS
lowCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':214, 'shortName':'tcc' }
midCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':224, 'shortName':'tcc' }
highCloudFrac_GFS =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':234, 'shortName':'tcc' }

#WWMCA
totalCloudFrac_WWMCA  = { 'parameterName':71, 'typeOfLevel':'entireAtmosphere', 'level':0 }

cloudTopHeightLev1_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':1 }
cloudTopHeightLev2_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':2 }
cloudTopHeightLev3_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':3 }
cloudTopHeightLev4_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':4 }
cloudTopHeight_WWMCA      = [ cloudTopHeightLev1_WWMCA, cloudTopHeightLev2_WWMCA,␣
↪cloudTopHeightLev3_WWMCA, cloudTopHeightLev4_WWMCA ]

cloudBaseHeightLev1_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':1 }
cloudBaseHeightLev2_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':2 }
cloudBaseHeightLev3_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':3 }
cloudBaseHeightLev4_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':4 }
cloudBaseHeight_WWMCA      = [ cloudBaseHeightLev1_WWMCA, cloudBaseHeightLev2_WWMCA,␣
↪cloudBaseHeightLev3_WWMCA, cloudBaseHeightLev4_WWMCA ]

verifVariablesModel = {
    'binaryCloud'    :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
    'totalCloudFrac' :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
    'lowCloudFrac'   :  {'GFS':[lowCloudFrac_GFS], 'GALWEM17':[lowCloudFrac_GALWEM], 'GALWEM
↪':[lowCloudFrac_GALWEM], 'MPAS':['cldfrac_low_UM']},
    'midCloudFrac'   :  {'GFS':[midCloudFrac_GFS], 'GALWEM17':[midCloudFrac_GALWEM], 'GALWEM
↪':[midCloudFrac_GALWEM], 'MPAS':['cldfrac_mid_UM']},
    'highCloudFrac'  :  {'GFS':[highCloudFrac_GFS], 'GALWEM17':[highCloudFrac_GALWEM],
↪'GALWEM':[highCloudFrac_GALWEM], 'MPAS':['cldfrac_high_UM']},
    'cloudTopHeight' :  {'GFS':['']               , 'GALWEM17':[cloudTopHeight_GALWEM],
↪'GALWEM':[cloudTopHeight_GALWEM], 'MPAS':['cldht_top_UM']},
    'cloudBaseHeight' : {'GFS':['']               , 'GALWEM17':[cloudBaseHeight_GALWEM],
```

(continues on next page)

(continued from previous page)

```
↪'GALWEM':[cloudBaseHeight_GALWEM], 'MPAS':['cldht_base_UM']},
}

cloudFracCatThresholds = '>0, <10.0, >=10.0, >=20.0, >=30.0, >=40.0, >=50.0, >=60.0, >=70.0,␣
↪>=80.0, >=90.0' # MET format string
brightnessTempThresholds = '<280.0, <275.0, <273.15, <270.0, <265.0, <260.0, <255.0, <250.0,
↪<245.0, <240.0, <235.0, <230.0, <225.0, <220.0, <215.0, <210.0, <=SFP1, <=SFP5, <=SFP10,
↪<=SFP25, <=SFP50, >=SFP50, >=SFP75, >=SFP90, >=SFP95, >=SFP99'
verifVariables = {
   'binaryCloud'    : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['TCC'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'NA',  'thresholds':'>0.0', 'interpMethod':'nearest' },
   'totalCloudFrac' : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['tcc'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin' },
   'lowCloudFrac'   : { 'MERRA2':['CLDLOW'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['lcc'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'midCloudFrac'   : { 'MERRA2':['CLDMID'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['MCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'highCloudFrac'  : { 'MERRA2':['CLDHGH'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['HCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'cloudTopTemp'   : { 'MERRA2':['CLDTMP'], 'SATCORPS':['cloud_temperature_top_level'],
↪'ERA5':['']   , 'units':'K',   'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopPres'   : { 'MERRA2':['CLDPRS'], 'SATCORPS':['cloud_pressure_top_level'],
↪'ERA5':['']   , 'units':'hPa', 'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopHeight' : { 'MERRA2':['']      , 'SATCORPS':['cloud_height_top_level'],
↪'ERA5':['']   , 'WWMCA':cloudTopHeight_WWMCA,  'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
   'cloudBaseHeight': { 'MERRA2':['']      , 'SATCORPS':['cloud_height_base_level'],
↪'ERA5':['cbh'], 'WWMCA':cloudBaseHeight_WWMCA, 'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
   'cloudCeiling'   : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'m',   'thresholds':'NA', 'interpMethod':'bilin'},
   'brightnessTemp' : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'K',   'thresholds':brightnessTempThresholds, 'interpMethod':'bilin
↪'},
}

# Combine the two dictionaries
# Only reason verifVariablesModel exists is just for space--verifVaribles gets too long if␣
↪we keep adding more datasets
for key in verifVariablesModel.keys():
```

(continues on next page)

(continued from previous page)

```
  x = verifVariablesModel[key]
  for key1 in x.keys():
     verifVariables[key][key1] = x[key1]

#f = '/glade/u/home/schwartz/cloud_verification/GFS_grib_0.25deg/2018112412/gfs.0p25.
↪2018112412.f006.grib2'
#grbs = pygrib.open(f)
#idx = pygrib.index(f,'parameterCategory','parameterNumber','typeOfFirstFixedSurface')
#model = 'GFS'
#variable = 'totCloudCover'
#x = verifVariablesModel[variable][model] # returns a list, whose ith element is a dictionary
# e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
#idx(parameterCategory=x[0]['parameterCategory'],parameterNumber=x[0]['parameterNumber'],
↪typeOfFirstFixedSurface=x[0]['typeOfFirstFixedSurface'])

# to read in an environmental variable
#x = os.getenv('a') # probably type string no matter what

###########

def getThreshold(variable):
   x = verifVariables[variable]['thresholds']
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getInterpMethod(variable):
   x = verifVariables[variable]['interpMethod'].upper()
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getTotalCloudFrac(source,data):
   if source == 'SATCORPS':
    # x = data[0][0,:,:,0] * 1.0E-2  # scaling
      x = (data[0][0,:,:,1]  + data[0][0,:,:,2] + data[0][0,:,:,3])*1.0E-2  # scaling
   #  y = data[0]
   #  x = np.sum( y[:,:,:,1:4],axis=3)
   elif source == 'MERRA2':
#      x = ( data[0][0,:,:]+data[1][0,:,:]+data[2][0,:,:] ) *100.0 # the ith element of data␣
↪is a numpy array
      x = data[0][0,:,:] * 100.0 # the ith element of data is a numpy array
      print(x.min(), x.max())
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
```

(continues on next page)

(continued from previous page)

```
      x = data[0][0,:,:] * 100.0
   elif source == 'SAT_WWMCA_MEAN':
      x = data[0][0,:,:] # already in %
   else:
      x = data[0]

   # This next line is WRONG.
   # Missing should be set to missing
   # Then, the non-missing values are 1s and 0s
   #output = np.where(x > 0.0, x, 0.0)
   #output = np.where(x < 0.0, -9999.0, x) # missing. currently used for SATCORPS

   x = np.where( x < 0.0  , 0.0,   x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%
   return x

def getBinaryCloud(source,data):
   y = getTotalCloudFrac(source,data)
   # keep NaNs as is, but then set everything else to either 100% or 0%
   x = np.where( np.isnan(y), y, np.where(y > 0.0, 100.0, 0.0) )
   return x

def getLayerCloudFrac(source,data,layer):
   if source == 'SATCORPS':
      if layer.lower().strip() == 'low'  : i = 1
      if layer.lower().strip() == 'mid'  : i = 2
      if layer.lower().strip() == 'high' : i = 3
      x = data[0][0,:,:,i] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 100.0
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
      x = data[0][0,:,:] * 100.0
   else:
      x = data[0]

   x = np.where( x < 0.0, 0.0, x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%

   return x

def getCloudTopTemp(source,data):
   if source == 'SATCORPS':
```

(continues on next page)

(continued from previous page)

```
      x = data[0][0,:,:,0] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
      x = data[0]
   return x

def getCloudTopPres(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E-1  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 1.0E-2  # scaling [Pa] -> [hPa]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
      x = data[0]
   return x

def getCloudTopHeight(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
         print('error with WWMCA Cloud top height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmax(tmp,axis=0) # get maximum cloud top height across all layers
   else:
      x = data[0]
```

(continues on next page)

(continued from previous page)

```
   # Eliminate unphysical values (assume cloud top shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y

def getCloudBaseHeight(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
         print('error with WWMCA Cloud base height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmin(tmp,axis=0) # get lowest cloud base over all layers
   else:
      x = data[0]

   # Eliminate unphysical values (assume cloud base shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y

def getCloudCeiling(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0]   #TBD
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] # TBD
      except: x = data[0][0,:,:]
   return x

# add other functions for different variables
```

(continues on next page)

(continued from previous page)

```
###########

def getDataArray(inputFile,source,variable,dataSource):
   # 1) inputFile:  File name--either observations or forecast
   # 2) source:     Obsevation source (e.g., MERRA, SATCORP, etc.)
   # 3) variable:   Variable to verify
   # 4) dataSource: If 1, process forecast file. If 2 process obs file.

#   # specifying names here temporarily. file names should be passed in to python from shell␣
↪script
#   if source == 'merra':      nc_file = '/gpfs/fs1/scratch/schwartz/MERRA/MERRA2_400.tavg1_
↪2d_rad_Nx.20181101.nc4'
#   elif source == 'satcorp':  nc_file = '/glade/scratch/bjung/met/test_satcorps/GEO-MRGD.
↪2018334.0000.GRID.NC'
#   elif source == 'era5':     nc_file = '/glade/scratch/bjung/met/test_era5/e5.oper.fc.sfc.
↪instan.128_164_tcc.regn320sc.2018111606_2018120112.nc'

   source = source.upper().strip()  # Force uppercase and get rid of blank spaces, for safety

   print('dataSource = ',dataSource)

   ftype = griddedDatasets[source]['ftype'].lower().strip()

   # Get file handle
   if ftype == 'nc':
      nc_fid = Dataset(inputFile, "r", format="NETCDF4")
      #nc_fid.set_auto_scale(True)
   elif ftype == 'grib':
      if source == 'WWMCA':
        idx = pygrib.index(inputFile,'parameterName','typeOfLevel','level')
      else:
        idx = pygrib.index(inputFile,'parameterCategory','parameterNumber',
↪'typeOfFirstFixedSurface')

   # dataSource == 1 means forecast, 2 means obs
#  if dataSource == 1: varsToRead = verifVariablesModel[variable][source] # if ftype == 'grib
↪', returns a list whose ith element is a dictionary. otherwise, just a list
#  if dataSource == 2: varsToRead = verifVariables[variable][source] # returns a list
   varsToRead = verifVariables[variable][source] # if ftype == 'grib', returns a list whose␣
↪ith element is a dictionary. otherwise, just a list

   print('Trying to read ',inputFile)

   # Get lat/lon information--currently not used
  #latVar = griddedDatasets[source]['latVar']
```

(continues on next page)

(continued from previous page)

```
  #lonVar = griddedDatasets[source]['lonVar']
  #lats = np.array(nc_fid.variables[latVar][:])   # extract/copy the data
  #lons = np.array(nc_fid.variables[lonVar][:] )

   #print(lats.max())
   #print(lons.max())

   # one way to deal with scale factors
   # probably using something like nc_fid.set_auto_scale(True) is better...
  #latMax = lats.max()
  #while latMax > 90.0:
  #   lons = lons * 0.1
  #   lats = lats * 0.1
  #   latMax = lats.max()

   # get data
   data = []
   for v in varsToRead:
      if ftype == 'grib':
         if source == 'WWMCA':
           x = idx(parameterName=v['parameterName'],typeOfLevel=v['typeOfLevel'],level=v[
↪'level'])[0] # by getting element 0, you get a pygrib message
         else:
            # e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
            if ( variable == 'cloudTopHeight' or variable == 'cloudBaseHeight') and source␣
↪== 'GALWEM17':
               x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[1] # by getting␣
↪element 1, you get a pygrib message
            else:
               x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[0] # by getting␣
↪element 0, you get a pygrib message
            if x.shortName != v['shortName']: print('Name mismatch!')
            #ADDED BY JOHN O
            print(x)
            print('Reading ', x.shortName, 'at level ', x.typeOfFirstFixedSurface)
         read_var = x.values # same x.data()[0]
         read_missing = x.missingValue
         print('missing value = ',read_missing)

         # The missing value (read_missing) for GALWEM17 and GALWEM cloud base/height is␣
↪9999, which is not the best choice because
         # those could be actual values. So we need to use the masked array part (below) to␣
↪handle which
```

(continues on next page)

(continued from previous page)

```
        # values are missing.  We also set read_missing to something unphysical to␣
↪essentially disable it.
        # Finally, if we don't change the 'missingValue' property in the GRIB2 file we are␣
↪eventually outputting,
        # the bitmap will get all messed up, because it will be based on 9999 instead of
↪$missing_values
        if variable == 'cloudTopHeight' or variable == 'cloudBaseHeight':
           read_missing = -9999.
           x['missingValue'] = read_missing
           if source == 'GALWEM17':
              #These are masked numpy arrays, with mask = True where there is a missing␣
↪value (no cloud)
              #Use np.ma.filled to create an ndarray where mask = True values are set to np.
↪nan
              read_var = np.ma.filled(read_var.astype(read_var.dtype), np.nan)
     elif ftype == 'nc':
        read_var = nc_fid.variables[v]         # extract/copy the data
        try:
           read_missing = read_var.missing_value  # get variable attributes. Each dataset␣
↪has own missing values.
        except:
           read_missing = -9999. # set a default missing value. probably only need to do␣
↪this for MPAS

     print('Reading ', v)

     this_var = np.array( read_var )        # to numpy array
    #print(read_missing, np.nan)
     this_var = np.where( this_var==read_missing, np.nan, this_var )
    #print(this_var.shape)
     data.append(this_var) # ith element of the list is a NUMPY ARRAY for the ith variable
    #print(type(this_var))
    #print(type(data))

  # Call a function to get the variable of interest.
  # Add a new function for each variable
  if variable == 'binaryCloud':     raw_data = getBinaryCloud(source,data)
  if variable == 'totalCloudFrac':  raw_data = getTotalCloudFrac(source,data)
  if variable == 'lowCloudFrac':    raw_data = getLayerCloudFrac(source,data,'low')
  if variable == 'midCloudFrac':    raw_data = getLayerCloudFrac(source,data,'mid')
  if variable == 'highCloudFrac':   raw_data = getLayerCloudFrac(source,data,'high')
  if variable == 'cloudTopTemp':    raw_data = getCloudTopTemp(source,data)
  if variable == 'cloudTopPres':    raw_data = getCloudTopPres(source,data)
  if variable == 'cloudTopHeight':  raw_data = getCloudTopHeight(source,data)
  if variable == 'cloudBaseHeight': raw_data = getCloudBaseHeight(source,data)
```

(continues on next page)

(continued from previous page)

```
   if variable == 'cloudCeiling':    raw_data = getCloudCeiling(source,data)

   raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

   # Array met_data is passed to MET
   # Graphics should plot $met_data to make sure things look correct
   if griddedDatasets[source]['flipY']:
      print('flipping ',source,' data about y-axis')
      met_data=np.flip(raw_data,axis=0).astype(float)
   else:
      met_data=raw_data.astype(float)

   # Make plotting optional or Just use plot_data_plane
#   plt_data=np.where(met_data<0, np.nan, met_data)
#   map=Basemap(projection='cyl',llcrnrlat=-90,urcrnrlat=90,llcrnrlon=-180,urcrnrlon=180,
↪resolution='c')
#   map.drawcoastlines()
#   map.drawcountries()
#   map.drawparallels(np.arange(-90,90,30),labels=[1,1,0,1])
#   map.drawmeridians(np.arange(0,360,60),labels=[1,1,0,1])
#   plt.contourf(lons,lats,plt_data,20,origin='upper',cmap=cm.Greens) #cm.gist_rainbow)
#   title=source+"_"+variable+"_"+str(validTime)
#   plt.title(title)
#   plt.colorbar(orientation='horizontal')
#   plt.savefig(title+".png")

   # If a forecast file, output a GRIB file with
   # 1 record containing the met_data
   # This is a hack, because right now, MET python embedding doesn't work with pygrib,
   #    so output the data to a temporary file, and then have MET read the temporary grib␣
↪file.
   # Starting with version 9.0 of MET, the hack isn't needed, and MET python embedding works␣
↪with pygrib
   outputFcstFile = False  # MUST be True for MET version < 9.0.  For MET 9.0+, optional
   if dataSource == 1 and ftype == 'grib':
      if outputFcstFile:
         grbtmp = x
         grbtmp['values']=met_data
         grbout = open('temp_fcst.grb2','ab')
         grbout.write(grbtmp.tostring())
         grbout.close() # Close the outfile GRIB file
         print('Successfully output temp_fcst.grb2')

   # Close files
```

(continues on next page)

(continued from previous page)

```
   if ftype == 'grib': idx.close()    # Close the input GRIB file
   if ftype == 'nc':   nc_fid.close() # Close the netCDF file

   return met_data

def obsError(fcstData,obsErrorFile,validDate,dataSource):

   print('Adding noise to the cloud fraction fields')
   print('Using obsErrorFile',obsErrorFile)

   # First load the obsError information
   #obsErrorFile = 'ob_errors.pk'
   infile = open(obsErrorFile,'rb')
   binEdges, binStddev = pk.load(infile) # 'numpy.ndarray' types
   infile.close()

   # Get 1d forecast data
   shape = fcstData.shape
   fcst = fcstData.flatten()

   # Set random number seed based on valid time and model
   if   dataSource.upper().strip() == 'MPAS':   ii = 10
   elif dataSource.upper().strip() == 'GALWEM': ii = 20
   elif dataSource.upper().strip() == 'GFS':    ii = 30
   np.random.seed(int(validDate*.1 + ii))

   # Find which bin the data is in
   for i in range(0,len(binEdges)-1):
      idx = np.where( (fcst >= binEdges[i]) & (fcst < binEdges[i+1]) )[0]
      n = len(idx) # number of points in the ith bin
      if n > 0: # check for empty bins
         randVals = np.random.normal(0,binStddev[i],n)
         fcst[idx] = fcst[idx] + randVals

   # bound forecast values to between 0 and 100%
   fcst = np.where( fcst < 0.0,     0.0,   fcst)
   fcst = np.where( fcst > 100.0,   100.0, fcst)

   # now reshape forecast data back to 2D
   output = fcst.reshape(shape)

   # data will have NaNs where bad.
   return output

def getFcstCloudFrac(cfr,pmid,psfc,layerDefinitions): # cfr is cloud fraction(%), pmid is 3D␣
```

(continues on next page)

(continued from previous page)

```
↪pressure(Pa), psfc is surface pressure (Pa) code from UPP ./INITPOST.F

   if pmid.shape != cfr.shape:  # sanity check
      print('dimension mismatch bewteen cldfra and pressure')
      sys.exit()

   nlocs, nlevs = pmid.shape

   if len(psfc) != nlocs: # another sanity check
      print('dimension mismatch bewteen cldfra and surface pressure')
      sys.exit()

   cfracl = np.zeros(nlocs)
   cfracm = np.zeros(nlocs)
   cfrach = np.zeros(nlocs)

   for i in range(0,nlocs):

      PTOP_HIGH = PTOP_HIGH_UPP
      if layerDefinitions.upper().strip() == 'ERA5':
         PTOP_LOW = 0.8*psfc[i]
         PTOP_MID = 0.45*psfc[i]
      elif layerDefinitions.upper().strip() == 'UPP':
         PTOP_LOW = PTOP_LOW_UPP
         PTOP_MID = PTOP_MID_UPP

      idxLow  = np.where(   pmid[i,:] >= PTOP_LOW)[0] # using np.where with just 1 argument␣
↪returns tuple
      idxMid  = np.where(  (pmid[i,:] <  PTOP_LOW) & (pmid[i,:] >= PTOP_MID))[0]
      idxHigh = np.where(  (pmid[i,:] <  PTOP_MID) & (pmid[i,:] >= PTOP_HIGH))[0]

      # use conditions in case all indices are missing
      if (len(idxLow) >0 ):  cfracl[i] = np.max( cfr[i,idxLow] )
      if (len(idxMid) >0 ):  cfracm[i] = np.max( cfr[i,idxMid] )
      if (len(idxHigh) >0 ): cfrach[i] = np.max( cfr[i,idxHigh] )

   tmp = np.vstack( (cfracl,cfracm,cfrach)) # stack the rows into one 2d array
   cldfraMax = np.max(tmp,axis=0) # get maximum value across low/mid/high for each pixel␣
↪(minimum overlap assumption)

   # This is the fortran code put into python format...double loop unnecessary and slow
   #for i in range(0,nlocs):
   #   for k in range(0,nlevs):
   #      if pmid(i,k) >= PTOP_LOW:
   #         cfracl(i) = np.max( [cfracl(i),cfr(i,k)] ) # Low
```

(continues on next page)

(continued from previous page)

```
    #      elif pmid(i,k) < PTOP_LOW and pmid(i,k) >= PTOP_MID:
    #         cfracm(i) = np.max( [cfracm(i),cfr(i,k)] ) # Mid
    #      elif pmid(i,k) < PTOP_MID and pmid(i,k) >= PTOP_HIGH: # High
    #         cfrach(i) = np.max( [cfrach(i),cfr(i,k)] )

    return cfracl, cfracm, cfrach, cldfraMax

def getGOES16LatLon(g16_data_file):

    # Start timer
    startTime = dt.datetime.utcnow()

    # designate dataset
    g16nc = Dataset(g16_data_file, 'r')

    # GOES-R projection info and retrieving relevant constants
    proj_info = g16nc.variables['goes_imager_projection']
    lon_origin = proj_info.longitude_of_projection_origin
    H = proj_info.perspective_point_height+proj_info.semi_major_axis
    r_eq = proj_info.semi_major_axis
    r_pol = proj_info.semi_minor_axis

    # Data info
    lat_rad_1d = g16nc.variables['x'][:]
    lon_rad_1d = g16nc.variables['y'][:]

    # close file when finished
    g16nc.close()
    g16nc = None

    # create meshgrid filled with radian angles
    lat_rad,lon_rad = np.meshgrid(lat_rad_1d,lon_rad_1d)

    # lat/lon calc routine from satellite radian angle vectors

    lambda_0 = (lon_origin*np.pi)/180.0

    a_var = np.power(np.sin(lat_rad),2.0) + (np.power(np.cos(lat_rad),2.0)*(np.power(np.
↪cos(lon_rad),2.0)+(((r_eq*r_eq)/(r_pol*r_pol))*np.power(np.sin(lon_rad),2.0))))
    b_var = -2.0*H*np.cos(lat_rad)*np.cos(lon_rad)
    c_var = (H**2.0)-(r_eq**2.0)

    r_s = (-1.0*b_var - np.sqrt((b_var**2)-(4.0*a_var*c_var)))/(2.0*a_var)

    s_x = r_s*np.cos(lat_rad)*np.cos(lon_rad)
```

(continues on next page)

(continued from previous page)

```
   s_y = - r_s*np.sin(lat_rad)
   s_z = r_s*np.cos(lat_rad)*np.sin(lon_rad)

   lat = (180.0/np.pi)*(np.arctan(((r_eq*r_eq)/(r_pol*r_pol))*((s_z/np.sqrt(((H-s_x)*(H-s_
↪x))+(s_y*s_y))))))
   lon = (lambda_0 - np.arctan(s_y/(H-s_x)))*(180.0/np.pi)

   # End timer
   endTime = dt.datetime.utcnow()
   time = (endTime - startTime).microseconds / (1000.0*1000.0)
   print('took %f4.1 seconds to get GOES16 lat/lon'%(time))

   return lon,lat # lat/lon are 2-d arrays

# --
def getGOESRetrivalData(goesFile,goesVar):

   if not os.path.exists(goesFile):
      print(goesFile+' not there. exit')
      sys.exit()

   # First get GOES lat/lon
   goesLon2d, goesLat2d = getGOES16LatLon(goesFile) # 2-d arrays
   goesLon = goesLon2d.flatten() # 1-d arrays
   goesLat = goesLat2d.flatten()

   # Now open the file and get the data we want
   nc_goes = Dataset(goesFile, "r", format="NETCDF4")

   # If the next line is true (it should be), this indicates the variable needs to be treated
   #  as an "unsigned 16-bit integer". This is a pain.  So we must use the "astype" method
   #  to change the variable type BEFORE applying scale_factor and add_offset.  After the␣
↪conversion
   #  we then can manually apply the scale factor and offset
   #goesVar = 'PRES'
   goesVar = goesVar.strip() # for safety
   if nc_goes.variables[goesVar]._Unsigned.lower().strip() == 'true':
      nc_goes.set_auto_scale(False) # Don't automatically apply scale_factor and add_offset␣
↪to variable
      goesData2d = np.array( nc_goes.variables[goesVar]).astype(np.uint16)
      goesData2d = goesData2d * nc_goes.variables[goesVar].scale_factor + nc_goes.
↪variables[goesVar].add_offset
      goesQC2d  = np.array( nc_goes.variables['DQF']).astype(np.uint8)
   else:
      goesData2d = np.array( nc_goes.variables[goesVar])
```

(continues on next page)

(continued from previous page)

```
      goesQC2d  = np.array( nc_goes.variables['DQF'])

   # Make variables 1-d
   goesQC  = goesQC2d.flatten()
   goesData = goesData2d.flatten()
   nc_goes.close()

   # Get rid of NaNs; base it on longitude
   goesData = goesData[~np.isnan(goesLon)] # Handle data arrays first before changing lat/
↪lon itself
   goesQC  = goesQC[~np.isnan(goesLon)]
   goesLon = goesLon[~np.isnan(goesLon)] # ~ is "logical not", also np.logical_not
   goesLat = goesLat[~np.isnan(goesLat)]
   if goesLon.shape != goesLat.shape:
      print('GOES lat/lon shape mismatch')
      sys.exit()

   # If goesQC == 0, good QC and there was a cloud with a valid pressure.
   # If goesQC == 4, no cloud; probably clear sky.
   # All other QC means no data, and we want to remove those points
   idx = np.logical_or( goesQC == 0, goesQC == 4) # Only keep QC == 0 or 4
   goesData = goesData[idx]
   goesQC  = goesQC[idx]
   goesLon = goesLon[idx]
   goesLat = goesLat[idx]

   # Only QC with 0 or 4 are left; now set QC == 4 to missing to indicate clear sky
   goesData = np.where( goesQC != 0, missing_values, goesData)

   # Get longitude to between (0,360) for consistency with JEDI files (this check is applied␣
↪to JEDI files, too)
   goesLon = np.where( goesLon < 0, goesLon + 360.0, goesLon )

   print('Min GOES Lon = ',np.min(goesLon))
   print('Max GOES Lon = ',np.max(goesLon))

   return goesLon, goesLat, goesData

def point2point(source,inputDir,satellite,channel,goesFile,condition,layerDefinitions,
↪dataSource):

   # Static Variables for QC and obs
   qcVar  = 'brightness_temperature_'+str(channel)+'@EffectiveQC' #'@EffectiveQC0' # QC␣
↪variable
   obsVar = 'brightness_temperature_'+str(channel)+'@ObsValue'  # Observation variable
```

(continues on next page)

(continued from previous page)

```
   # Get GOES-16 retrieval file with auxiliary information
   if 'abi' in satellite or 'ahi' in satellite:
      goesLon, goesLat, goesData = getGOESRetrivalData(goesFile,'PRES') # return 1-d arrays
      lonlatGOES = np.array( list(zip(goesLon, goesLat))) # lon/lat pairs for each GOES ob␣
↪(nobs_GOES, 2)
     #print('shape lonlatGOES = ',lonlatGOES.shape)
      print('getting data from ',goesFile)
      myGOESInterpolator = NearestNDInterpolator(lonlatGOES,goesData)

   # First check to see if there's a concatenated file with all obs.
   #  If so, use that.  If not, have to process one file per processor, which takes a lot␣
↪more time
   if os.path.exists(inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'):
      inputFiles =  [inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'] # needs to be in a list␣
↪since we loop over inputFiles
   else:
      # Get list of OMB files to process.  There is one file per processor.
      # Need to get them in order so they are called in the same order for the
      # forecast and observed passes through this subroutine.
      files = os.listdir(inputDir)
      inputFiles = fnmatch.filter(files,'obsout*_'+satellite+'*nc4') # returns relative path␣
↪names
      inputFiles = [inputDir+'/'+s for s in inputFiles] # add on directory name
      inputFiles.sort() # Get in order from low to high
   if len(inputFiles) == 0: return -99999, -99999 # if no matching files, force a failure

   # Variable to pull for brightness temperature
#  if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@GsiHofXBc' # Forecast␣
↪variable
   if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@hofx' #'@depbg' # OMB
   if dataSource == 2: v = obsVar

   # Read the files and put data in array
   allData, allDataQC = [], []
   for inputFile in inputFiles:
      nc_fid = Dataset(inputFile, "r", format="NETCDF4") #Dataset is the class behavior to␣
↪open the file
      print('Trying to read ',v,' from ',inputFile)

      # Read forecast/obs data
      read_var = nc_fid.variables[v]         # extract/copy the data
   #  read_missing = read_var.missing_value  # get variable attributes. Each dataset has own␣
↪missing values.
      this_var = np.array( read_var )        # to numpy array
```

(continues on next page)

(continued from previous page)

```
  #  this_var = np.where( this_var==read_missing, np.nan, this_var )

    #if dataSource == 1: # If true, we just read in OMB data, but we want B
    #   obsData = np.array( nc_fid.variables[obsVar])
    #   this_var = obsData - this_var # get background/forecast value (O - OMB = B)

     #Read QC data
     qcData = np.array(nc_fid.variables[qcVar])

     # Sanity check...shapes should match
     if qcData.shape != this_var.shape: return -99999, -99999

     if 'abi' in satellite or 'ahi' in satellite:

        # Get the GOES-16 retrieval data at the observation locations in this file
        #   GOES values < 0 mean clear sky
        lats = np.array(nc_fid.variables['latitude@MetaData'])
        lons = np.array(nc_fid.variables['longitude@MetaData'])

        # Get longitude to between (0,360) for consistency with GOES-16 files
        lons = np.where( lons < 0, lons + 360.0, lons )

        lonlat = np.array( list(zip(lons,lats)))  # lon/lat pairs for each ob (nobs, 2)
        thisGOESData = myGOESInterpolator(lonlat) # GOES data at obs locations in this file.
↪ If pressure, units are hPa
        thisGOESData = thisGOESData * 100.0 # get into Pa

        #obsCldfra = np.array( nc_fid.variables['cloud_area_fraction@MetaData'] )*100.0 #␣
↪Get into %...observed cloud fraction (AHI/ABI only)

        geoValsFile = inputFile.replace('obsout','geoval')
        if not os.path.exists(geoValsFile):
           print(geoValsFile+' not there. exit')
           sys.exit()

        nc_fid2 = Dataset(geoValsFile, "r", format="NETCDF4")
        fcstCldfra = np.array( nc_fid2.variables['cloud_area_fraction_in_atmosphere_layer
↪'])*100.0 # Get into %
        pressure   = np.array( nc_fid2.variables['air_pressure']) # Pa
        pressure_edges   = np.array( nc_fid2.variables['air_pressure_levels']) # Pa
        psfc = pressure_edges[:,-1]  # Surface pressure (Pa)...array order is top down
        if layerDefinitions.upper().strip() == 'ERA5':
           PTOP_LOW = 0.8*psfc # these are arrays
           PTOP_MID = 0.45*psfc
           PTOP_HIGH = PTOP_HIGH_UPP * np.ones_like(psfc)
```

(continues on next page)

(continued from previous page)

```
        elif layerDefinitions.upper().strip() == 'UPP':
           PTOP_LOW = PTOP_LOW_UPP # these are constants
           PTOP_MID = PTOP_MID_UPP
           PTOP_HIGH = PTOP_HIGH_UPP
        else:
           print('layerDefinitions = ',layerDefinitions,'is invalid. exit')
           sys.exit()
        fcstLow,fcstMid,fcstHigh,fcstTotCldFra = getFcstCloudFrac(fcstCldfra,pressure,psfc,
↪layerDefinitions) # get low/mid/high/total forecast cloud fractions for each ob
        nc_fid2.close()

        # Modify QC data based on correspondence between forecast and obs. qcData used to␣
↪select good data later
        # It's possible that there are multiple forecast layers, such that fcstLow,fcstMid,
↪fcstHigh are all > $cldfraThresh
        # However, GOES-16 CTP doesn't really account for layering.  So, we need to remove␣
↪layered clouds from the forecast,
        #   focusing only on the layers that we asked for when doing {low,mid,high}Only␣
↪conditions
        # The "|" is symbol for "np.logcal_or"
        yes = 2.0
        no  = 0.0
        cldfraThresh = 20.0 # percent
        if qcData.shape == fcstTotCldFra.shape == thisGOESData.shape:  # these should all␣
↪match
           print('Using condition ',condition,'for ABI/AHI')

           # Note that "&" is "np.logical_and" for boolean (true/false) quantities.
           # Thus, each condition should be enclosed in parentheses
           if   condition.lower().strip() == 'clearOnly'.lower():  # clear in both forecast␣
↪and obs
              qcData = np.where( (fcstTotCldFra < cldfraThresh)  & (thisGOESData <= 0.0),␣
↪qcData, missing_values)
           elif condition.lower().strip() == 'cloudyOnly'.lower(): # cloudy in both␣
↪forecast and obs
              qcData = np.where( (fcstTotCldFra >= cldfraThresh) & (thisGOESData > 0.0),␣
↪qcData, missing_values)
           elif condition.lower().strip() == 'lowOnly'.lower(): # low clouds in both␣
↪forecast and obs
              fcstLow = np.where( (fcstMid >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstLow) # remove mid, high
              qcData = np.where( (fcstLow >= cldfraThresh) & ( thisGOESData >= PTOP_LOW),␣
↪qcData, missing_values)
           elif condition.lower().strip() == 'midOnly'.lower(): # mid clouds in both␣
↪forecast and obs
```

(continues on next page)

(continued from previous page)

```
            fcstMid = np.where( (fcstLow >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstMid) # remove low, high
            qcData = np.where( (fcstMid >= cldfraThresh) & (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID),   qcData, missing_values)
         elif condition.lower().strip() == 'highOnly'.lower(): # high clouds in both␣
↪forecast and obs
            fcstHigh = np.where( (fcstLow >= cldfraThresh) | ( fcstMid >= cldfraThresh),␣
↪missing_values, fcstHigh) # remove mid, high
            qcData = np.where( (fcstHigh >= cldfraThresh) & (thisGOESData <  PTOP_MID) &␣
↪(thisGOESData >= PTOP_HIGH), qcData, missing_values)
         elif condition.lower().strip() == 'fcstLow'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
            qcData = np.where( fcstLow >= cldfraThresh , qcData, missing_values)
         elif condition.lower().strip() == 'fcstMid'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
            qcData = np.where( fcstMid >= cldfraThresh , qcData, missing_values)
         elif condition.lower().strip() == 'fcstHigh'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
            qcData = np.where( fcstHigh >= cldfraThresh , qcData, missing_values)
         elif condition.lower().strip() == 'cloudEventLow'.lower():
            if dataSource == 1: this_var = np.where( fcstLow      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
            if dataSource == 2: this_var = np.where( thisGOESData >= PTOP_LOW, yes, no )
         elif condition.lower().strip() == 'cloudEventMid'.lower():
            if dataSource == 1: this_var = np.where( fcstMid      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
            if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID), yes, no )
         elif condition.lower().strip() == 'cloudEventHigh'.lower():
            if dataSource == 1: this_var = np.where( fcstHigh     >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
            if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_MID) &␣
↪(thisGOESData >= PTOP_HIGH), yes, no )
         elif condition.lower().strip() == 'cloudEventTot'.lower():
            if dataSource == 1: this_var = np.where( fcstTotCldFra >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
            if dataSource == 2: this_var = np.where( thisGOESData  > 0.0, yes, no )
         elif condition.lower().strip() == 'all':
            print("not doing any conditional verification or stratifying by event")
         else:
            print("condition = ",condition," not recognized.")
            sys.exit()
         #elif condition.lower().strip() == '4x4table'.lower():
           #if dataSource == 1:
           #   this_var = np.where( fcstLow >= cldfraThresh, yesLow, no )
```

(continues on next page)

(continued from previous page)

```
            #   this_var = this_var + np.where( fcstMid >= cldfraThresh, yesMid, no )
            #   this_var = this_var + np.where( fcstHigh >= cldfraThresh, yesHigh, no )
          print('number removed = ', (qcData==missing_values).sum())
         #print('number passed   = ', qcData.shape[0] - (qcData==missing_values).sum())
       else:
          print('shape mismatch')
          return -99999, -99999

    # Append to arrays
    allData.append(this_var)
    allDataQC.append(qcData)

    nc_fid.close() # done with the file, so close it before going to next file in loop

  # We're now all done looping over the individul files

  # Get the indices with acceptable QC
  allQC = np.concatenate(allDataQC) # Put list of numpy arrays into a single long 1-D numpy␣
↪array.  All QC data.
  idx = np.where(allQC==0) # returns indices

  # Now get all the forecast/observed brightness temperature data with acceptable QC
  this_var = np.concatenate(allData)[idx] # Put list of numpy arrays into a single long 1-D␣
↪numpy array. This is all the forecast/obs data with good QC
  numObs = this_var.shape[0] # number of points with good QC for this channel
  print('Number of obs :',numObs)

  # Assume all the points actually fit into a square grid. Get the side of the square (use␣
↪ceil to round up)
  if numObs > 0:
     l = np.ceil(np.sqrt(numObs)).astype('int') # Length of the side of the square

     # Make an array that can be reshaped into the square
     raw_data1D = np.full(l*l,np.nan) # Initialize 1D array of length l**2 to np.nan
     raw_data1D[0:numObs] = this_var[:] # Fill data to the extent possible. There will be␣
↪some np.nan values at the end
     raw_data = np.reshape(raw_data1D,(l,l)) # Reshape into "square grid"

     raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

     met_data=raw_data.astype(float) # Give MET this info

     # Now need to tell MET the "grid" for the data
     # Make a fake lat/lon grid going from 0.0 to 50.0 degrees, with the interval␣
```

(continues on next page)

(continued from previous page)

```
↪determined by number of points
      griddedDatasets[source]['latDef'][0] = 0.0 # starting point
      griddedDatasets[source]['latDef'][1] = np.diff(np.linspace(0,50,l)).round(6)[0] #␣
↪interval (degrees)
      griddedDatasets[source]['latDef'][2] = int(l) # number of points
      griddedDatasets[source]['lonDef'][0:3] = griddedDatasets[source]['latDef']

      gridInfo = getGridInfo(source, griddedDatasets[source]['gridType']) # 'LatLon' gridType
      return met_data, gridInfo

   else:
      return -99999, -99999

###########
def getGridInfo(source,gridType):

   if gridType == 'LatLon':
      latDef = griddedDatasets[source]['latDef']
      lonDef = griddedDatasets[source]['lonDef']
      gridInfo = {
         'type':      gridType,
         'name':      source,
         'lat_ll':    latDef[0], #-90.000,
         'lon_ll':    lonDef[0], #-180.000,
         'delta_lat': latDef[1], #0.5000,
         'delta_lon': lonDef[1], #0.625,
         'Nlat':      latDef[2], #361,
         'Nlon':      lonDef[2], #576,
      }
   elif gridType == 'Gaussian':
      gridInfo = {
        'type':     gridType,
        'name':     source,
        'nx':       griddedDatasets[source]['nx'],
        'ny':       griddedDatasets[source]['ny'],
        'lon_zero': griddedDatasets[source]['lon_zero'],
      }

   return gridInfo

def getAttrArray(source,variable,initTime,validTime):

   init = dt.datetime.strptime(initTime,"%Y%m%d%H")
   valid = dt.datetime.strptime(validTime,"%Y%m%d%H")
   lead, rem = divmod((valid-init).total_seconds(), 3600)
```

(continues on next page)

(continued from previous page)

```
   attrs = {

      'valid': valid.strftime("%Y%m%d_%H%M%S"),
      'init':  init.strftime("%Y%m%d_%H%M%S"),
      'lead':  str(int(lead)),
      'accum': '000000',

      'name':      variable,  #'MERRA2_Cloud_Percentage'
      'long_name': variable,  #'Cloud Percentage Levels',
      'level':     'ALL',
      'units':     verifVariables[variable]['units'],

      'grid': getGridInfo(source,griddedDatasets[source]['gridType'])
   }

   #print(attrs)
   #print(griddedDatasets[source])

   return attrs

######## END FUNCTIONS   ##########


#if __name__ == "__main__":
dataFile, dataSource, variable, i_date, v_date, flag = sys.argv[1].split(":")
met_data = getDataArray(dataFile,dataSource,variable,flag)
attrs = getAttrArray(dataSource,variable,i_date,v_date)
print(attrs)
```

Running METplus

Pass the use case configuration file to the run_metplus.py script along with any user-specific system configuration files if desired:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/clouds/GridStat_fcstGFS_
↪obsERA5_lowAndTotalCloudFrac.conf /path/to/user_system.conf
```

See *Running METplus* (page 26) for more information.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/clouds/GridStat_fcstGFS_obsERA5_lowAndTotalCloudFrac (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_GFS_to_ERA5_F36_CloudFracs_360000L_20220705_000000V_pairs.nc
- grid_stat_GFS_to_ERA5_F36_CloudFracs_360000L_20220705_000000V.stat
- grid_stat_GFS_to_ERA5_F36_CloudFracs_NBR_360000L_20220705_000000V_pairs.nc
- grid_stat_GFS_to_ERA5_F36_CloudFracs_NBR_360000L_20220705_000000V.stat
- grid_stat_GFS_to_ERA5_F36_CloudFracs_PROB_360000L_20220705_000000V_pairs.nc
- grid_stat_GFS_to_ERA5_F36_CloudFracs_PRB_360000L_20220705_000000V.stat

Keywords

Note:

- GridStatToolUseCase
- NetCDFFileUseCase
- CloudsAppUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/clouds-GridStat_fcstGFS_obsERA5_lowAndTotalCloudFrac.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.3.2 GridStat: Cloud Height with Neighborhood and Probabilities

model_applications/clouds/GridStat_fcstMPAS_obsERA5_cloudBaseHgt.conf

Scientific Objective

This use case captures various statistical measures of two model comparisons for cloud base height with different neighborhood settings for internal model metrics and to aid in future model updates

Datasets

Forecast: Model for Prediction Across Scales (MPAS)
Observations: ECMWF Reanalysis, Version 5 (ERA5)
Grid: GPP 17km masking region

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

METplus Components

This use case utilizes Python Embedding, which is called using the PYTHON_NUMPY keyword in the forecast and observation input template settings. The same Python script processes both forecast and observation datasets. The forecast field is verified against the respective observation field, with the Python script being passed the input file, the model name, the variable name being analyzed, the initialization and valid times, and a flag to indicate if the field passed is observation or forecast. This process is repeated with 2 instance names to GridStat, each with a different setting for regridding, neighborhood evaluation, thresholding, output line types, and output prefix names.

METplus Workflow

GridStat is the only MET tool called in this example. It processes the following run time:

Init: 2020-07-23 00Z
Forecast lead: 36 hour

Because instance names are used, GridStat will run 2 times for this 1 initalization time.

METplus Configuration

METplus first loads the default configuration file found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line: parm/use_cases/model_applications/clouds/GridStat_fcstMPAS_obsERA5_cloudBaseHgt.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/clouds/GridStat_
↪fcstMPAS_obsERA5_cloudBaseHgt.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

# ###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###


PROCESS_LIST = GridStat, GridStat(nbr)

###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2020072300
INIT_END=2020072300
INIT_INCREMENT = 12H

LEAD_SEQ = 36

LOOP_ORDER = times

###
# File I/O
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR =
FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_GRID_STAT_INPUT_DIR =
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_CLIMO_MEAN_INPUT_DIR =
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/clouds/GridStat_fcstMPAS_obsERA5_
↪cloudBaseHgt
GRID_STAT_OUTPUT_TEMPLATE =


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = MPAS
OBTYPE = ERA5

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/clouds/GridStat_fcstMPAS_obsERA5_
↪cloudBaseHgt

FCST_VAR1_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstMPAS_obsERA5_cloudBaseHgt/diag.{valid?fmt=%Y-%m-%d_%H}.00.00_latlon.nc:{MODEL}
↪:cloudBaseHeight:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:1
FCST_VAR1_LEVELS =
FCST_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0

OBS_VAR1_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstMPAS_obsERA5_cloudBaseHgt/ERA5_{valid?fmt=%Y%m%d%H}_Cld.nc:{OBTYPE}
↪:cloudBaseHeight:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:2
OBS_VAR1_LEVELS =
OBS_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0
```

(continues on next page)

(continued from previous page)

```
###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

#LOG_GRID_STAT_VERBOSITY = 2

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

GRID_STAT_REGRID_TO_GRID = FCST
GRID_STAT_REGRID_METHOD = BILIN
GRID_STAT_REGRID_WIDTH = 2

GRID_STAT_DESC =

FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0
OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_ONCE_PER_FIELD = False

FCST_IS_PROB = false

FCST_GRID_STAT_PROB_THRESH = ==0.1

OBS_IS_PROB = false

OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudBaseHght

GRID_STAT_OUTPUT_FLAG_FHO = STAT
GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_CNT = STAT
GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = STAT
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_MASK_POLY = {INPUT_BASE}/model_applications/clouds/GridStat_fcstMPAS_obsERA5_
↪cloudBaseHgt/GPP_17km_60S_60N_mask.nc

[nbr]

FCST_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SFP20, >SFP30, >SFP40, >SFP50, >SFP60, >SFP70, >SFP80

OBS_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SOP20, >SOP30, >SOP40, >SOP50, >SOP60, >SOP70, >SOP80

GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9
GRID_STAT_NEIGHBORHOOD_SHAPE = CIRCLE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >0.0

GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = NONE
GRID_STAT_OUTPUT_FLAG_CTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = NONE
GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTC = STAT
GRID_STAT_OUTPUT_FLAG_NBRCTS = STAT
GRID_STAT_OUTPUT_FLAG_NBRCNT = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = NONE

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_NBRHD = TRUE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudBaseHght_NBR
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
```

(continues on next page)

(continued from previous page)

```
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
```

(continues on next page)

(continued from previous page)

```
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case utilizes 1 Python script to read and process both forecast and observation fields. parm/use_cases/model_applications/clouds/GridStat_fcstMPAS_obsERA5_cloudBaseHgt/read_input_data.py

```
#this code was provided by Craig Schwartz
#and is largely unaltered from its original
#function.

#from __future__ import print_function
import os
import sys
import numpy as np
import datetime as dt
from netCDF4 import Dataset  # http://code.google.com/p/netcdf4-python/
```

(continues on next page)

(continued from previous page)

```
from scipy.interpolate import NearestNDInterpolator, LinearNDInterpolator
#### for Plotting
import matplotlib.cm as cm
import matplotlib.axes as maxes
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
#from mpl_toolkits.basemap import Basemap
import fnmatch
import pygrib
import pickle as pk
#####

##########################################

missing_values = -9999.0  # for MET

# UPP top layer bounds (Pa) for cloud layers
PTOP_LOW_UPP  = 64200. # low for > 64200 Pa
PTOP_MID_UPP  = 35000. # mid between 35000-64200 Pa
PTOP_HIGH_UPP = 15000. # high between 15000-35000 Pa

# Values for 4 x 4 contingency table
Na, Nb, Nc, Nd = 1, 2, 3, 4
Ne, Nf, Ng, Nh = 5, 6, 7, 8
Ni, Nj, Nk, Nl = 9, 10, 11, 12
Nm, Nn, No, Np = 13, 14, 15, 16

# Notes:
# 1) Entry for 'point' is for point-to-point comparison and is all dummy data (except for␣
↪gridType) that is overwritten by point2point
# 2) ERA5 on NCAR CISL RDA changed at some point.  Old is ERA5_2017 (not used anymore), new␣
↪is ERA5, which we'll use for 2020 data
griddedDatasets =  {
   'MERRA2'    : { 'gridType':'LatLon', 'latVar':'lat',      'latDef':[-90.0,0.50,361], 'lonVar
↪':'lon',        'lonDef':[-180.0,0.625,576],   'flipY':True, 'ftype':'nc'},
   'SATCORPS' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[-180.0,0.3125,1152], 'flipY':False, 'ftype':'nc' },
   'ERA5_2017': { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.7848769072,0.
↪281016829130516,640], 'lonVar':'longitude', 'lonDef':[0.0,0.28125,1280], 'flipY':False,
↪'ftype':'nc' },
   'ERA5'      : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'GFS'       : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'GALWEM'    : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
```

(continues on next page)

(continued from previous page)

```
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':True, 'ftype':'grib'},
   'GALWEM17' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.921875,0.156250,
↪1152], 'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536], 'flipY':False, 'ftype':
↪'grib'},
   'WWMCA'    : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'MPAS'     : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721],
↪'lonVar':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'nc'},
   'SAT_WWMCA_MEAN' : { 'gridType':'LatLon', 'latVar':'lat','latDef':[-90.0,0.25,721],
↪'lonVar':'lon', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'point'    : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.156250,1152],
↪'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536],   'flipY':False, 'ftype':'nc'},
}
   #TODO:Correct one, but MET can ingest a Gaussian grid only in Grib2 format (from Randy B.)
   #'ERA5'     : { 'gridType':'Gaussian', 'nx':1280, 'ny':640, 'lon_zero':0, 'latVar':
↪'latitude', 'lonVar':'longitude', 'flipY':False, },

#GALWEM, both 17-km and 0.25-degree
lowCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':3,
↪'typeOfFirstFixedSurface':10, 'shortName':'lcc' }
midCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':4,
↪'typeOfFirstFixedSurface':10, 'shortName':'mcc' }
highCloudFrac_GALWEM =  { 'parameterCategory':6, 'parameterNumber':5,
↪'typeOfFirstFixedSurface':10, 'shortName':'hcc' }
totalCloudFrac_GALWEM = { 'parameterCategory':6, 'parameterNumber':1,
↪'typeOfFirstFixedSurface':10, 'shortName':'tcc' }
cloudTopHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':12,
↪'typeOfFirstFixedSurface':3, 'shortName':'cdct' }
cloudBaseHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':11,
↪'typeOfFirstFixedSurface':2, 'shortName':'cdcb' }

#GFS
lowCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':214, 'shortName':'tcc' }
midCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':224, 'shortName':'tcc' }
highCloudFrac_GFS =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':234, 'shortName':'tcc' }

#WWMCA
totalCloudFrac_WWMCA  = { 'parameterName':71, 'typeOfLevel':'entireAtmosphere', 'level':0 }

cloudTopHeightLev1_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':1 }
cloudTopHeightLev2_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':2 }
cloudTopHeightLev3_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':3 }
```

(continues on next page)

(continued from previous page)

```
cloudTopHeightLev4_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':4 }
cloudTopHeight_WWMCA      = [ cloudTopHeightLev1_WWMCA, cloudTopHeightLev2_WWMCA,␣
↪cloudTopHeightLev3_WWMCA, cloudTopHeightLev4_WWMCA ]

cloudBaseHeightLev1_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':1 }
cloudBaseHeightLev2_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':2 }
cloudBaseHeightLev3_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':3 }
cloudBaseHeightLev4_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':4 }
cloudBaseHeight_WWMCA      = [ cloudBaseHeightLev1_WWMCA, cloudBaseHeightLev2_WWMCA,␣
↪cloudBaseHeightLev3_WWMCA, cloudBaseHeightLev4_WWMCA ]

verifVariablesModel = {
    'binaryCloud'    :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
    'totalCloudFrac' :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
    'lowCloudFrac'   :  {'GFS':[lowCloudFrac_GFS], 'GALWEM17':[lowCloudFrac_GALWEM], 'GALWEM
↪':[lowCloudFrac_GALWEM], 'MPAS':['cldfrac_low_UM']},
    'midCloudFrac'   :  {'GFS':[midCloudFrac_GFS], 'GALWEM17':[midCloudFrac_GALWEM], 'GALWEM
↪':[midCloudFrac_GALWEM], 'MPAS':['cldfrac_mid_UM']},
    'highCloudFrac'  :  {'GFS':[highCloudFrac_GFS], 'GALWEM17':[highCloudFrac_GALWEM],
↪'GALWEM':[highCloudFrac_GALWEM], 'MPAS':['cldfrac_high_UM']},
    'cloudTopHeight' :  {'GFS':['']               , 'GALWEM17':[cloudTopHeight_GALWEM],
↪'GALWEM':[cloudTopHeight_GALWEM], 'MPAS':['cldht_top_UM']},
    'cloudBaseHeight' : {'GFS':['']               , 'GALWEM17':[cloudBaseHeight_GALWEM],
↪'GALWEM':[cloudBaseHeight_GALWEM], 'MPAS':['cldht_base_UM']},
}

cloudFracCatThresholds = '>0, <10.0, >=10.0, >=20.0, >=30.0, >=40.0, >=50.0, >=60.0, >=70.0,␣
↪>=80.0, >=90.0' # MET format string
brightnessTempThresholds = '<280.0, <275.0, <273.15, <270.0, <265.0, <260.0, <255.0, <250.0,
↪<245.0, <240.0, <235.0, <230.0, <225.0, <220.0, <215.0, <210.0, <=SFP1, <=SFP5, <=SFP10,
↪<=SFP25, <=SFP50, >=SFP50, >=SFP75, >=SFP90, >=SFP95, >=SFP99'
verifVariables = {
   'binaryCloud'    : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['TCC'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'NA',  'thresholds':'>0.0', 'interpMethod':'nearest' },
   'totalCloudFrac' : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['tcc'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin' },
   'lowCloudFrac'   : { 'MERRA2':['CLDLOW'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['lcc'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'midCloudFrac'   : { 'MERRA2':['CLDMID'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['MCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
```

(continues on next page)

(continued from previous page)

```
↪},
   'highCloudFrac'  : { 'MERRA2':['CLDHGH'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['HCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'cloudTopTemp'   : { 'MERRA2':['CLDTMP'], 'SATCORPS':['cloud_temperature_top_level'],
↪'ERA5':['']   , 'units':'K',   'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopPres'   : { 'MERRA2':['CLDPRS'], 'SATCORPS':['cloud_pressure_top_level'],
↪'ERA5':['']   , 'units':'hPa', 'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopHeight' : { 'MERRA2':['']      , 'SATCORPS':['cloud_height_top_level'],
↪'ERA5':['']   , 'WWMCA':cloudTopHeight_WWMCA,  'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
   'cloudBaseHeight': { 'MERRA2':['']      , 'SATCORPS':['cloud_height_base_level'],
↪'ERA5':['cbh'], 'WWMCA':cloudBaseHeight_WWMCA, 'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
   'cloudCeiling'   : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'m',   'thresholds':'NA', 'interpMethod':'bilin'},
   'brightnessTemp' : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'K',   'thresholds':brightnessTempThresholds, 'interpMethod':'bilin
↪'},
}

# Combine the two dictionaries
# Only reason verifVariablesModel exists is just for space--verifVaribles gets too long if␣
↪we keep adding more datasets
for key in verifVariablesModel.keys():
  x = verifVariablesModel[key]
  for key1 in x.keys():
     verifVariables[key][key1] = x[key1]

#f = '/glade/u/home/schwartz/cloud_verification/GFS_grib_0.25deg/2018112412/gfs.0p25.
↪2018112412.f006.grib2'
#grbs = pygrib.open(f)
#idx = pygrib.index(f,'parameterCategory','parameterNumber','typeOfFirstFixedSurface')
#model = 'GFS'
#variable = 'totCloudCover'
#x = verifVariablesModel[variable][model] # returns a list, whose ith element is a dictionary
# e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
#idx(parameterCategory=x[0]['parameterCategory'],parameterNumber=x[0]['parameterNumber'],
↪typeOfFirstFixedSurface=x[0]['typeOfFirstFixedSurface'])

# to read in an environmental variable
#x = os.getenv('a') # probably type string no matter what

###########
```

(continues on next page)

(continued from previous page)

```
def getThreshold(variable):
   x = verifVariables[variable]['thresholds']
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getInterpMethod(variable):
   x = verifVariables[variable]['interpMethod'].upper()
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getTotalCloudFrac(source,data):
   if source == 'SATCORPS':
    # x = data[0][0,:,:,0] * 1.0E-2  # scaling
      x = (data[0][0,:,:,1]  + data[0][0,:,:,2] + data[0][0,:,:,3])*1.0E-2  # scaling
   #  y = data[0]
   #  x = np.sum( y[:,:,:,1:4],axis=3)
   elif source == 'MERRA2':
#      x = ( data[0][0,:,:]+data[1][0,:,:]+data[2][0,:,:] ) *100.0 # the ith element of data␣
↪is a numpy array
      x = data[0][0,:,:] * 100.0 # the ith element of data is a numpy array
      print(x.min(), x.max())
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
      x = data[0][0,:,:] * 100.0
   elif source == 'SAT_WWMCA_MEAN':
      x = data[0][0,:,:] # already in %
   else:
      x = data[0]

   # This next line is WRONG.
   # Missing should be set to missing
   # Then, the non-missing values are 1s and 0s
   #output = np.where(x > 0.0, x, 0.0)
   #output = np.where(x < 0.0, -9999.0, x) # missing. currently used for SATCORPS

   x = np.where( x < 0.0  , 0.0,   x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%
   return x

def getBinaryCloud(source,data):
   y = getTotalCloudFrac(source,data)
   # keep NaNs as is, but then set everything else to either 100% or 0%
   x = np.where( np.isnan(y), y, np.where(y > 0.0, 100.0, 0.0) )
```

(continues on next page)

(continued from previous page)

```
   return x

def getLayerCloudFrac(source,data,layer):
   if source == 'SATCORPS':
      if layer.lower().strip() == 'low'  : i = 1
      if layer.lower().strip() == 'mid'  : i = 2
      if layer.lower().strip() == 'high' : i = 3
      x = data[0][0,:,:,i] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 100.0
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
      x = data[0][0,:,:] * 100.0
   else:
      x = data[0]

   x = np.where( x < 0.0, 0.0, x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%

   return x

def getCloudTopTemp(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
      x = data[0]
   return x

def getCloudTopPres(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E-1  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 1.0E-2  # scaling [Pa] -> [hPa]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
      x = data[0]
```

(continues on next page)

(continued from previous page)

```
   return x

def getCloudTopHeight(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
         print('error with WWMCA Cloud top height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmax(tmp,axis=0) # get maximum cloud top height across all layers
   else:
      x = data[0]

   # Eliminate unphysical values (assume cloud top shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y

def getCloudBaseHeight(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
```

(continues on next page)

(continued from previous page)

```
         print('error with WWMCA Cloud base height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmin(tmp,axis=0) # get lowest cloud base over all layers
   else:
      x = data[0]

   # Eliminate unphysical values (assume cloud base shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y

def getCloudCeiling(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0]   #TBD
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] # TBD
      except: x = data[0][0,:,:]
   return x

# add other functions for different variables

###########

def getDataArray(inputFile,source,variable,dataSource):
   # 1) inputFile:  File name--either observations or forecast
   # 2) source:     Obsevation source (e.g., MERRA, SATCORP, etc.)
   # 3) variable:   Variable to verify
   # 4) dataSource: If 1, process forecast file. If 2 process obs file.

#   # specifying names here temporarily. file names should be passed in to python from shell␣
↪script
#   if source == 'merra':      nc_file = '/gpfs/fs1/scratch/schwartz/MERRA/MERRA2_400.tavg1_
↪2d_rad_Nx.20181101.nc4'
#   elif source == 'satcorp':  nc_file = '/glade/scratch/bjung/met/test_satcorps/GEO-MRGD.
↪2018334.0000.GRID.NC'
#   elif source == 'era5':     nc_file = '/glade/scratch/bjung/met/test_era5/e5.oper.fc.sfc.
↪instan.128_164_tcc.regn320sc.2018111606_2018120112.nc'

   source = source.upper().strip()  # Force uppercase and get rid of blank spaces, for safety

   print('dataSource = ',dataSource)
```

(continues on next page)

(continued from previous page)

```
   ftype = griddedDatasets[source]['ftype'].lower().strip()

   # Get file handle
   if ftype == 'nc':
      nc_fid = Dataset(inputFile, "r", format="NETCDF4")
      #nc_fid.set_auto_scale(True)
   elif ftype == 'grib':
      if source == 'WWMCA':
        idx = pygrib.index(inputFile,'parameterName','typeOfLevel','level')
      else:
        idx = pygrib.index(inputFile,'parameterCategory','parameterNumber',
↪'typeOfFirstFixedSurface')

   # dataSource == 1 means forecast, 2 means obs
#  if dataSource == 1: varsToRead = verifVariablesModel[variable][source] # if ftype == 'grib
↪', returns a list whose ith element is a dictionary. otherwise, just a list
#  if dataSource == 2: varsToRead = verifVariables[variable][source] # returns a list
   varsToRead = verifVariables[variable][source] # if ftype == 'grib', returns a list whose␣
↪ith element is a dictionary. otherwise, just a list

   print('Trying to read ',inputFile)

   # Get lat/lon information--currently not used
  #latVar = griddedDatasets[source]['latVar']
  #lonVar = griddedDatasets[source]['lonVar']
  #lats = np.array(nc_fid.variables[latVar][:])   # extract/copy the data
  #lons = np.array(nc_fid.variables[lonVar][:] )

   #print(lats.max())
   #print(lons.max())

   # one way to deal with scale factors
   # probably using something like nc_fid.set_auto_scale(True) is better...
  #latMax = lats.max()
  #while latMax > 90.0:
  #   lons = lons * 0.1
  #   lats = lats * 0.1
  #   latMax = lats.max()

   # get data
   data = []
   for v in varsToRead:
      if ftype == 'grib':
         if source == 'WWMCA':
```

(continues on next page)

(continued from previous page)

```
          x = idx(parameterName=v['parameterName'],typeOfLevel=v['typeOfLevel'],level=v[
↪'level'])[0] # by getting element 0, you get a pygrib message
        else:
           # e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
           if ( variable == 'cloudTopHeight' or variable == 'cloudBaseHeight') and source␣
↪== 'GALWEM17':
              x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[1] # by getting␣
↪element 1, you get a pygrib message
           else:
              x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[0] # by getting␣
↪element 0, you get a pygrib message
           if x.shortName != v['shortName']: print('Name mismatch!')
           #ADDED BY JOHN O
           print(x)
           print('Reading ', x.shortName, 'at level ', x.typeOfFirstFixedSurface)
        read_var = x.values # same x.data()[0]
        read_missing = x.missingValue
        print('missing value = ',read_missing)

        # The missing value (read_missing) for GALWEM17 and GALWEM cloud base/height is␣
↪9999, which is not the best choice because
        # those could be actual values. So we need to use the masked array part (below) to␣
↪handle which
        # values are missing.  We also set read_missing to something unphysical to␣
↪essentially disable it.
        # Finally, if we don't change the 'missingValue' property in the GRIB2 file we are␣
↪eventually outputting,
        # the bitmap will get all messed up, because it will be based on 9999 instead of
↪$missing_values
        if variable == 'cloudTopHeight' or variable == 'cloudBaseHeight':
           read_missing = -9999.
           x['missingValue'] = read_missing
           if source == 'GALWEM17':
              #These are masked numpy arrays, with mask = True where there is a missing␣
↪value (no cloud)
              #Use np.ma.filled to create an ndarray where mask = True values are set to np.
↪nan
              read_var = np.ma.filled(read_var.astype(read_var.dtype), np.nan)
     elif ftype == 'nc':
        read_var = nc_fid.variables[v]         # extract/copy the data
        try:
           read_missing = read_var.missing_value  # get variable attributes. Each dataset␣
↪has own missing values.
```

(continues on next page)

(continued from previous page)

```
        except:
           read_missing = -9999. # set a default missing value. probably only need to do␣
↪this for MPAS

     print('Reading ', v)

     this_var = np.array( read_var )        # to numpy array
    #print(read_missing, np.nan)
     this_var = np.where( this_var==read_missing, np.nan, this_var )
    #print(this_var.shape)
     data.append(this_var) # ith element of the list is a NUMPY ARRAY for the ith variable
    #print(type(this_var))
    #print(type(data))

  # Call a function to get the variable of interest.
  # Add a new function for each variable
  if variable == 'binaryCloud':     raw_data = getBinaryCloud(source,data)
  if variable == 'totalCloudFrac':  raw_data = getTotalCloudFrac(source,data)
  if variable == 'lowCloudFrac':    raw_data = getLayerCloudFrac(source,data,'low')
  if variable == 'midCloudFrac':    raw_data = getLayerCloudFrac(source,data,'mid')
  if variable == 'highCloudFrac':   raw_data = getLayerCloudFrac(source,data,'high')
  if variable == 'cloudTopTemp':    raw_data = getCloudTopTemp(source,data)
  if variable == 'cloudTopPres':    raw_data = getCloudTopPres(source,data)
  if variable == 'cloudTopHeight':  raw_data = getCloudTopHeight(source,data)
  if variable == 'cloudBaseHeight': raw_data = getCloudBaseHeight(source,data)
  if variable == 'cloudCeiling':    raw_data = getCloudCeiling(source,data)

  raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

  # Array met_data is passed to MET
  # Graphics should plot $met_data to make sure things look correct
  if griddedDatasets[source]['flipY']:
     print('flipping ',source,' data about y-axis')
     met_data=np.flip(raw_data,axis=0).astype(float)
  else:
     met_data=raw_data.astype(float)

  # Make plotting optional or Just use plot_data_plane
#   plt_data=np.where(met_data<0, np.nan, met_data)
#   map=Basemap(projection='cyl',llcrnrlat=-90,urcrnrlat=90,llcrnrlon=-180,urcrnrlon=180,
↪resolution='c')
#   map.drawcoastlines()
#   map.drawcountries()
#   map.drawparallels(np.arange(-90,90,30),labels=[1,1,0,1])
```

(continues on next page)

(continued from previous page)

```
#   map.drawmeridians(np.arange(0,360,60),labels=[1,1,0,1])
#   plt.contourf(lons,lats,plt_data,20,origin='upper',cmap=cm.Greens) #cm.gist_rainbow)
#   title=source+"_"+variable+"_"+str(validTime)
#   plt.title(title)
#   plt.colorbar(orientation='horizontal')
#   plt.savefig(title+".png")

   # If a forecast file, output a GRIB file with
   # 1 record containing the met_data
   # This is a hack, because right now, MET python embedding doesn't work with pygrib,
   #    so output the data to a temporary file, and then have MET read the temporary grib␣
↪file.
   # Starting with version 9.0 of MET, the hack isn't needed, and MET python embedding works␣
↪with pygrib
   outputFcstFile = False  # MUST be True for MET version < 9.0.  For MET 9.0+, optional
   if dataSource == 1 and ftype == 'grib':
      if outputFcstFile:
         grbtmp = x
         grbtmp['values']=met_data
         grbout = open('temp_fcst.grb2','ab')
         grbout.write(grbtmp.tostring())
         grbout.close() # Close the outfile GRIB file
         print('Successfully output temp_fcst.grb2')

   # Close files
   if ftype == 'grib': idx.close()    # Close the input GRIB file
   if ftype == 'nc':   nc_fid.close() # Close the netCDF file

   return met_data

def obsError(fcstData,obsErrorFile,validDate,dataSource):

   print('Adding noise to the cloud fraction fields')
   print('Using obsErrorFile',obsErrorFile)

   # First load the obsError information
   #obsErrorFile = 'ob_errors.pk'
   infile = open(obsErrorFile,'rb')
   binEdges, binStddev = pk.load(infile) # 'numpy.ndarray' types
   infile.close()

   # Get 1d forecast data
   shape = fcstData.shape
   fcst = fcstData.flatten()
```

(continues on next page)

(continued from previous page)

```
   # Set random number seed based on valid time and model
   if   dataSource.upper().strip() == 'MPAS':   ii = 10
   elif dataSource.upper().strip() == 'GALWEM': ii = 20
   elif dataSource.upper().strip() == 'GFS':    ii = 30
   np.random.seed(int(validDate*.1 + ii))

   # Find which bin the data is in
   for i in range(0,len(binEdges)-1):
      idx = np.where( (fcst >= binEdges[i]) & (fcst < binEdges[i+1]) )[0]
      n = len(idx) # number of points in the ith bin
      if n > 0: # check for empty bins
         randVals = np.random.normal(0,binStddev[i],n)
         fcst[idx] = fcst[idx] + randVals

   # bound forecast values to between 0 and 100%
   fcst = np.where( fcst < 0.0,     0.0,   fcst)
   fcst = np.where( fcst > 100.0,   100.0, fcst)

   # now reshape forecast data back to 2D
   output = fcst.reshape(shape)

   # data will have NaNs where bad.
   return output

def getFcstCloudFrac(cfr,pmid,psfc,layerDefinitions): # cfr is cloud fraction(%), pmid is 3D␣
↪pressure(Pa), psfc is surface pressure (Pa) code from UPP ./INITPOST.F

   if pmid.shape != cfr.shape:  # sanity check
      print('dimension mismatch bewteen cldfra and pressure')
      sys.exit()

   nlocs, nlevs = pmid.shape

   if len(psfc) != nlocs: # another sanity check
      print('dimension mismatch bewteen cldfra and surface pressure')
      sys.exit()

   cfracl = np.zeros(nlocs)
   cfracm = np.zeros(nlocs)
   cfrach = np.zeros(nlocs)

   for i in range(0,nlocs):

      PTOP_HIGH = PTOP_HIGH_UPP
      if layerDefinitions.upper().strip() == 'ERA5':
```

(continues on next page)

(continued from previous page)

```
         PTOP_LOW = 0.8*psfc[i]
         PTOP_MID = 0.45*psfc[i]
      elif layerDefinitions.upper().strip() == 'UPP':
         PTOP_LOW = PTOP_LOW_UPP
         PTOP_MID = PTOP_MID_UPP

      idxLow  = np.where(   pmid[i,:] >= PTOP_LOW)[0] # using np.where with just 1 argument␣
↪returns tuple
      idxMid  = np.where(  (pmid[i,:] <  PTOP_LOW) & (pmid[i,:] >= PTOP_MID))[0]
      idxHigh = np.where(  (pmid[i,:] <  PTOP_MID) & (pmid[i,:] >= PTOP_HIGH))[0]

      # use conditions in case all indices are missing
      if (len(idxLow) >0 ):  cfracl[i] = np.max( cfr[i,idxLow] )
      if (len(idxMid) >0 ):  cfracm[i] = np.max( cfr[i,idxMid] )
      if (len(idxHigh) >0 ): cfrach[i] = np.max( cfr[i,idxHigh] )

   tmp = np.vstack( (cfracl,cfracm,cfrach)) # stack the rows into one 2d array
   cldfraMax = np.max(tmp,axis=0) # get maximum value across low/mid/high for each pixel␣
↪(minimum overlap assumption)

   # This is the fortran code put into python format...double loop unnecessary and slow
   #for i in range(0,nlocs):
   #   for k in range(0,nlevs):
   #      if pmid(i,k) >= PTOP_LOW:
   #         cfracl(i) = np.max( [cfracl(i),cfr(i,k)] ) # Low
   #      elif pmid(i,k) < PTOP_LOW and pmid(i,k) >= PTOP_MID:
   #         cfracm(i) = np.max( [cfracm(i),cfr(i,k)] ) # Mid
   #      elif pmid(i,k) < PTOP_MID and pmid(i,k) >= PTOP_HIGH: # High
   #         cfrach(i) = np.max( [cfrach(i),cfr(i,k)] )

   return cfracl, cfracm, cfrach, cldfraMax

def getGOES16LatLon(g16_data_file):

   # Start timer
   startTime = dt.datetime.utcnow()

   # designate dataset
   g16nc = Dataset(g16_data_file, 'r')

   # GOES-R projection info and retrieving relevant constants
   proj_info = g16nc.variables['goes_imager_projection']
   lon_origin = proj_info.longitude_of_projection_origin
   H = proj_info.perspective_point_height+proj_info.semi_major_axis
   r_eq = proj_info.semi_major_axis
```

(continues on next page)

(continued from previous page)

```
   r_pol = proj_info.semi_minor_axis

   # Data info
   lat_rad_1d = g16nc.variables['x'][:]
   lon_rad_1d = g16nc.variables['y'][:]

   # close file when finished
   g16nc.close()
   g16nc = None

   # create meshgrid filled with radian angles
   lat_rad,lon_rad = np.meshgrid(lat_rad_1d,lon_rad_1d)

   # lat/lon calc routine from satellite radian angle vectors

   lambda_0 = (lon_origin*np.pi)/180.0

   a_var = np.power(np.sin(lat_rad),2.0) + (np.power(np.cos(lat_rad),2.0)*(np.power(np.
↪cos(lon_rad),2.0)+(((r_eq*r_eq)/(r_pol*r_pol))*np.power(np.sin(lon_rad),2.0))))
   b_var = -2.0*H*np.cos(lat_rad)*np.cos(lon_rad)
   c_var = (H**2.0)-(r_eq**2.0)

   r_s = (-1.0*b_var - np.sqrt((b_var**2)-(4.0*a_var*c_var)))/(2.0*a_var)

   s_x = r_s*np.cos(lat_rad)*np.cos(lon_rad)
   s_y = - r_s*np.sin(lat_rad)
   s_z = r_s*np.cos(lat_rad)*np.sin(lon_rad)

   lat = (180.0/np.pi)*(np.arctan(((r_eq*r_eq)/(r_pol*r_pol))*((s_z/np.sqrt(((H-s_x)*(H-s_
↪x))+(s_y*s_y))))))
   lon = (lambda_0 - np.arctan(s_y/(H-s_x)))*(180.0/np.pi)

   # End timer
   endTime = dt.datetime.utcnow()
   time = (endTime - startTime).microseconds / (1000.0*1000.0)
   print('took %f4.1 seconds to get GOES16 lat/lon'%(time))

   return lon,lat # lat/lon are 2-d arrays

# --
def getGOESRetrivalData(goesFile,goesVar):

   if not os.path.exists(goesFile):
      print(goesFile+' not there. exit')
      sys.exit()
```

(continues on next page)

(continued from previous page)

```
   # First get GOES lat/lon
   goesLon2d, goesLat2d = getGOES16LatLon(goesFile) # 2-d arrays
   goesLon = goesLon2d.flatten() # 1-d arrays
   goesLat = goesLat2d.flatten()

   # Now open the file and get the data we want
   nc_goes = Dataset(goesFile, "r", format="NETCDF4")

   # If the next line is true (it should be), this indicates the variable needs to be treated
   #  as an "unsigned 16-bit integer". This is a pain.  So we must use the "astype" method
   #  to change the variable type BEFORE applying scale_factor and add_offset.  After the␣
↪conversion
   #  we then can manually apply the scale factor and offset
   #goesVar = 'PRES'
   goesVar = goesVar.strip() # for safety
   if nc_goes.variables[goesVar]._Unsigned.lower().strip() == 'true':
      nc_goes.set_auto_scale(False) # Don't automatically apply scale_factor and add_offset␣
↪to variable
      goesData2d = np.array( nc_goes.variables[goesVar]).astype(np.uint16)
      goesData2d = goesData2d * nc_goes.variables[goesVar].scale_factor + nc_goes.
↪variables[goesVar].add_offset
      goesQC2d  = np.array( nc_goes.variables['DQF']).astype(np.uint8)
   else:
      goesData2d = np.array( nc_goes.variables[goesVar])
      goesQC2d  = np.array( nc_goes.variables['DQF'])

   # Make variables 1-d
   goesQC  = goesQC2d.flatten()
   goesData = goesData2d.flatten()
   nc_goes.close()

   # Get rid of NaNs; base it on longitude
   goesData = goesData[~np.isnan(goesLon)] # Handle data arrays first before changing lat/
↪lon itself
   goesQC  = goesQC[~np.isnan(goesLon)]
   goesLon = goesLon[~np.isnan(goesLon)] # ~ is "logical not", also np.logical_not
   goesLat = goesLat[~np.isnan(goesLat)]
   if goesLon.shape != goesLat.shape:
      print('GOES lat/lon shape mismatch')
      sys.exit()

   # If goesQC == 0, good QC and there was a cloud with a valid pressure.
   # If goesQC == 4, no cloud; probably clear sky.
   # All other QC means no data, and we want to remove those points
```

(continues on next page)

(continued from previous page)

```
   idx = np.logical_or( goesQC == 0, goesQC == 4) # Only keep QC == 0 or 4
   goesData = goesData[idx]
   goesQC  = goesQC[idx]
   goesLon = goesLon[idx]
   goesLat = goesLat[idx]

   # Only QC with 0 or 4 are left; now set QC == 4 to missing to indicate clear sky
   goesData = np.where( goesQC != 0, missing_values, goesData)

   # Get longitude to between (0,360) for consistency with JEDI files (this check is applied␣
↪to JEDI files, too)
   goesLon = np.where( goesLon < 0, goesLon + 360.0, goesLon )

   print('Min GOES Lon = ',np.min(goesLon))
   print('Max GOES Lon = ',np.max(goesLon))

   return goesLon, goesLat, goesData

def point2point(source,inputDir,satellite,channel,goesFile,condition,layerDefinitions,
↪dataSource):

   # Static Variables for QC and obs
   qcVar  = 'brightness_temperature_'+str(channel)+'@EffectiveQC' #'@EffectiveQC0' # QC␣
↪variable
   obsVar = 'brightness_temperature_'+str(channel)+'@ObsValue'  # Observation variable

   # Get GOES-16 retrieval file with auxiliary information
   if 'abi' in satellite or 'ahi' in satellite:
      goesLon, goesLat, goesData = getGOESRetrivalData(goesFile,'PRES') # return 1-d arrays
      lonlatGOES = np.array( list(zip(goesLon, goesLat))) # lon/lat pairs for each GOES ob␣
↪(nobs_GOES, 2)
     #print('shape lonlatGOES = ',lonlatGOES.shape)
      print('getting data from ',goesFile)
      myGOESInterpolator = NearestNDInterpolator(lonlatGOES,goesData)

   # First check to see if there's a concatenated file with all obs.
   #  If so, use that.  If not, have to process one file per processor, which takes a lot␣
↪more time
   if os.path.exists(inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'):
      inputFiles =  [inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'] # needs to be in a list␣
↪since we loop over inputFiles
   else:
      # Get list of OMB files to process.  There is one file per processor.
      # Need to get them in order so they are called in the same order for the
      # forecast and observed passes through this subroutine.
```

(continues on next page)

(continued from previous page)

```
      files = os.listdir(inputDir)
      inputFiles = fnmatch.filter(files,'obsout*_'+satellite+'*nc4') # returns relative path␣
↪names
      inputFiles = [inputDir+'/'+s for s in inputFiles] # add on directory name
      inputFiles.sort() # Get in order from low to high
   if len(inputFiles) == 0: return -99999, -99999 # if no matching files, force a failure

   # Variable to pull for brightness temperature
#  if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@GsiHofXBc' # Forecast␣
↪variable
   if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@hofx' #'@depbg' # OMB
   if dataSource == 2: v = obsVar

   # Read the files and put data in array
   allData, allDataQC = [], []
   for inputFile in inputFiles:
      nc_fid = Dataset(inputFile, "r", format="NETCDF4") #Dataset is the class behavior to␣
↪open the file
      print('Trying to read ',v,' from ',inputFile)

      # Read forecast/obs data
      read_var = nc_fid.variables[v]         # extract/copy the data
   #  read_missing = read_var.missing_value  # get variable attributes. Each dataset has own␣
↪missing values.
      this_var = np.array( read_var )        # to numpy array
   #  this_var = np.where( this_var==read_missing, np.nan, this_var )

     #if dataSource == 1: # If true, we just read in OMB data, but we want B
     #   obsData = np.array( nc_fid.variables[obsVar])
     #   this_var = obsData - this_var # get background/forecast value (O - OMB = B)

      #Read QC data
      qcData = np.array(nc_fid.variables[qcVar])

      # Sanity check...shapes should match
      if qcData.shape != this_var.shape: return -99999, -99999

      if 'abi' in satellite or 'ahi' in satellite:

         # Get the GOES-16 retrieval data at the observation locations in this file
         #   GOES values < 0 mean clear sky
         lats = np.array(nc_fid.variables['latitude@MetaData'])
         lons = np.array(nc_fid.variables['longitude@MetaData'])

         # Get longitude to between (0,360) for consistency with GOES-16 files
```

(continues on next page)

(continued from previous page)

```
        lons = np.where( lons < 0, lons + 360.0, lons )

        lonlat = np.array( list(zip(lons,lats)))  # lon/lat pairs for each ob (nobs, 2)
        thisGOESData = myGOESInterpolator(lonlat) # GOES data at obs locations in this file.
↪ If pressure, units are hPa
        thisGOESData = thisGOESData * 100.0 # get into Pa

        #obsCldfra = np.array( nc_fid.variables['cloud_area_fraction@MetaData'] )*100.0 #␣
↪Get into %...observed cloud fraction (AHI/ABI only)

        geoValsFile = inputFile.replace('obsout','geoval')
        if not os.path.exists(geoValsFile):
           print(geoValsFile+' not there. exit')
           sys.exit()

        nc_fid2 = Dataset(geoValsFile, "r", format="NETCDF4")
        fcstCldfra = np.array( nc_fid2.variables['cloud_area_fraction_in_atmosphere_layer
↪'])*100.0 # Get into %
        pressure   = np.array( nc_fid2.variables['air_pressure']) # Pa
        pressure_edges   = np.array( nc_fid2.variables['air_pressure_levels']) # Pa
        psfc = pressure_edges[:,-1]  # Surface pressure (Pa)...array order is top down
        if layerDefinitions.upper().strip() == 'ERA5':
           PTOP_LOW = 0.8*psfc # these are arrays
           PTOP_MID = 0.45*psfc
           PTOP_HIGH = PTOP_HIGH_UPP * np.ones_like(psfc)
        elif layerDefinitions.upper().strip() == 'UPP':
           PTOP_LOW = PTOP_LOW_UPP # these are constants
           PTOP_MID = PTOP_MID_UPP
           PTOP_HIGH = PTOP_HIGH_UPP
        else:
           print('layerDefinitions = ',layerDefinitions,'is invalid. exit')
           sys.exit()
        fcstLow,fcstMid,fcstHigh,fcstTotCldFra = getFcstCloudFrac(fcstCldfra,pressure,psfc,
↪layerDefinitions) # get low/mid/high/total forecast cloud fractions for each ob
        nc_fid2.close()

        # Modify QC data based on correspondence between forecast and obs. qcData used to␣
↪select good data later
        # It's possible that there are multiple forecast layers, such that fcstLow,fcstMid,
↪fcstHigh are all > $cldfraThresh
        # However, GOES-16 CTP doesn't really account for layering.  So, we need to remove␣
↪layered clouds from the forecast,
        #   focusing only on the layers that we asked for when doing {low,mid,high}Only␣
↪conditions
        # The "|" is symbol for "np.logcal_or"
```

(continues on next page)

(continued from previous page)

```
        yes = 2.0
        no  = 0.0
        cldfraThresh = 20.0 # percent
        if qcData.shape == fcstTotCldFra.shape == thisGOESData.shape:  # these should all␣
↪match
            print('Using condition ',condition,'for ABI/AHI')

            # Note that "&" is "np.logical_and" for boolean (true/false) quantities.
            # Thus, each condition should be enclosed in parentheses
            if   condition.lower().strip() == 'clearOnly'.lower():  # clear in both forecast␣
↪and obs
                qcData = np.where( (fcstTotCldFra < cldfraThresh)  & (thisGOESData <= 0.0),␣
↪qcData, missing_values)
            elif condition.lower().strip() == 'cloudyOnly'.lower(): # cloudy in both␣
↪forecast and obs
                qcData = np.where( (fcstTotCldFra >= cldfraThresh) & (thisGOESData > 0.0),␣
↪qcData, missing_values)
            elif condition.lower().strip() == 'lowOnly'.lower(): # low clouds in both␣
↪forecast and obs
                fcstLow = np.where( (fcstMid >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstLow) # remove mid, high
                qcData = np.where( (fcstLow >= cldfraThresh) & ( thisGOESData >= PTOP_LOW),␣
↪qcData, missing_values)
            elif condition.lower().strip() == 'midOnly'.lower(): # mid clouds in both␣
↪forecast and obs
                fcstMid = np.where( (fcstLow >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstMid) # remove low, high
                qcData = np.where( (fcstMid >= cldfraThresh) & (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID),   qcData, missing_values)
            elif condition.lower().strip() == 'highOnly'.lower(): # high clouds in both␣
↪forecast and obs
                fcstHigh = np.where( (fcstLow >= cldfraThresh) | ( fcstMid >= cldfraThresh),␣
↪missing_values, fcstHigh) # remove mid, high
                qcData = np.where( (fcstHigh >= cldfraThresh) & (thisGOESData <  PTOP_MID) &␣
↪(thisGOESData >= PTOP_HIGH), qcData, missing_values)
            elif condition.lower().strip() == 'fcstLow'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
                qcData = np.where( fcstLow >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'fcstMid'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
                qcData = np.where( fcstMid >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'fcstHigh'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
                qcData = np.where( fcstHigh >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'cloudEventLow'.lower():
```

(continues on next page)

(continued from previous page)

```
               if dataSource == 1: this_var = np.where( fcstLow      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
               if dataSource == 2: this_var = np.where( thisGOESData >= PTOP_LOW, yes, no )
            elif condition.lower().strip() == 'cloudEventMid'.lower():
               if dataSource == 1: this_var = np.where( fcstMid      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
               if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID), yes, no )
            elif condition.lower().strip() == 'cloudEventHigh'.lower():
               if dataSource == 1: this_var = np.where( fcstHigh     >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
               if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_MID) &␣
↪(thisGOESData >= PTOP_HIGH), yes, no )
            elif condition.lower().strip() == 'cloudEventTot'.lower():
               if dataSource == 1: this_var = np.where( fcstTotCldFra >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
               if dataSource == 2: this_var = np.where( thisGOESData  > 0.0, yes, no )
            elif condition.lower().strip() == 'all':
               print("not doing any conditional verification or stratifying by event")
            else:
               print("condition = ",condition," not recognized.")
               sys.exit()
            #elif condition.lower().strip() == '4x4table'.lower():
              #if dataSource == 1:
              #   this_var = np.where( fcstLow >= cldfraThresh, yesLow, no )
              #   this_var = this_var + np.where( fcstMid >= cldfraThresh, yesMid, no )
              #   this_var = this_var + np.where( fcstHigh >= cldfraThresh, yesHigh, no )
            print('number removed = ', (qcData==missing_values).sum())
           #print('number passed   = ', qcData.shape[0] - (qcData==missing_values).sum())
         else:
            print('shape mismatch')
            return -99999, -99999

      # Append to arrays
      allData.append(this_var)
      allDataQC.append(qcData)

      nc_fid.close() # done with the file, so close it before going to next file in loop

   # We're now all done looping over the individul files

   # Get the indices with acceptable QC
   allQC = np.concatenate(allDataQC) # Put list of numpy arrays into a single long 1-D numpy␣
↪array.  All QC data.
   idx = np.where(allQC==0) # returns indices
```

(continues on next page)

(continued from previous page)

```
   # Now get all the forecast/observed brightness temperature data with acceptable QC
   this_var = np.concatenate(allData)[idx] # Put list of numpy arrays into a single long 1-D␣
↪numpy array. This is all the forecast/obs data with good QC
   numObs = this_var.shape[0] # number of points with good QC for this channel
   print('Number of obs :',numObs)

   # Assume all the points actually fit into a square grid. Get the side of the square (use␣
↪ceil to round up)
   if numObs > 0:
      l = np.ceil(np.sqrt(numObs)).astype('int') # Length of the side of the square

      # Make an array that can be reshaped into the square
      raw_data1D = np.full(l*l,np.nan) # Initialize 1D array of length l**2 to np.nan
      raw_data1D[0:numObs] = this_var[:] # Fill data to the extent possible. There will be␣
↪some np.nan values at the end
      raw_data = np.reshape(raw_data1D,(l,l)) # Reshape into "square grid"

      raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

      met_data=raw_data.astype(float) # Give MET this info

      # Now need to tell MET the "grid" for the data
      # Make a fake lat/lon grid going from 0.0 to 50.0 degrees, with the interval␣
↪determined by number of points
      griddedDatasets[source]['latDef'][0] = 0.0 # starting point
      griddedDatasets[source]['latDef'][1] = np.diff(np.linspace(0,50,l)).round(6)[0] #␣
↪interval (degrees)
      griddedDatasets[source]['latDef'][2] = int(l) # number of points
      griddedDatasets[source]['lonDef'][0:3] = griddedDatasets[source]['latDef']

      gridInfo = getGridInfo(source, griddedDatasets[source]['gridType']) # 'LatLon' gridType
      return met_data, gridInfo

   else:
      return -99999, -99999

###########
def getGridInfo(source,gridType):

   if gridType == 'LatLon':
      latDef = griddedDatasets[source]['latDef']
      lonDef = griddedDatasets[source]['lonDef']
      gridInfo = {
```

(continues on next page)

(continued from previous page)

```
        'type':      gridType,
        'name':      source,
        'lat_ll':    latDef[0], #-90.000,
        'lon_ll':    lonDef[0], #-180.000,
        'delta_lat': latDef[1], #0.5000,
        'delta_lon': lonDef[1], #0.625,
        'Nlat':      latDef[2], #361,
        'Nlon':      lonDef[2], #576,
     }
  elif gridType == 'Gaussian':
     gridInfo = {
       'type':     gridType,
       'name':     source,
       'nx':       griddedDatasets[source]['nx'],
       'ny':       griddedDatasets[source]['ny'],
       'lon_zero': griddedDatasets[source]['lon_zero'],
     }

  return gridInfo

def getAttrArray(source,variable,initTime,validTime):

  init = dt.datetime.strptime(initTime,"%Y%m%d%H")
  valid = dt.datetime.strptime(validTime,"%Y%m%d%H")
  lead, rem = divmod((valid-init).total_seconds(), 3600)

  attrs = {

     'valid': valid.strftime("%Y%m%d_%H%M%S"),
     'init':  init.strftime("%Y%m%d_%H%M%S"),
     'lead':  str(int(lead)),
     'accum': '000000',

     'name':      variable,  #'MERRA2_Cloud_Percentage'
     'long_name': variable,  #'Cloud Percentage Levels',
     'level':     'ALL',
     'units':     verifVariables[variable]['units'],

     'grid': getGridInfo(source,griddedDatasets[source]['gridType'])
  }

  #print(attrs)
  #print(griddedDatasets[source])

  return attrs
```

(continues on next page)

(continued from previous page)

```
######## END FUNCTIONS   ##########


#if __name__ == "__main__":
dataFile, dataSource, variable, i_date, v_date, flag = sys.argv[1].split(":")
met_data = getDataArray(dataFile,dataSource,variable,flag)
attrs = getAttrArray(dataSource,variable,i_date,v_date)
print(attrs)
```

Running METplus

Pass the use case configuration file to the run_metplus.py script along with any user-specific system configuration files if desired:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/clouds/GridStat_fcstMPAS_
↪obsERA5_cloudBaseHgt.conf /path/to/user_system.conf
```

See *Running METplus* (page 26) for more information.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/clouds/GridStat_fcstMPAS_obsERA5_cloudBaseHgt (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_MPAS_to_ERA5_F36_CloudBaseHgt_360000L_20200724_120000V_pairs.nc
- grid_stat_MPAS_to_ERA5_F36_CloudBaseHgt_360000L_20200724_120000V.stat
- grid_stat_MPAS_to_ERA5_F36_CloudBaseHgt_NBR_360000L_20200724_120000V_pairs.nc
- grid_stat_MPAS_to_ERA5_F36_CloudBaseHgt_NBR_360000L_20200724_120000V.stat

Keywords

Note:

- GridStatToolUseCase
- NetCDFFileUseCase
- CloudsAppUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/clouds-GridStat_fcstMPAS_obsERA5_cloudBaseHgt.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.3.3 GridStat: Cloud Pressure and Temperature Heights

model_applications/clouds/GridStat_fcstGFS_obsSATCORPS_cloudTopPressAndTemp.conf

Scientific Objective

This use case captures various statistical measures of two model comparisons for cloud top pressures and temperatures with different neighborhood settings for internal model metrics and to aid in future model updates

Datasets

Forecast: Global Forecast System (GFS)
Observations: Satellite ClOud and Radiation Property retrieval System (SatCORPS)
Grid: GPP 17km masking region

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

METplus Components

This use case utilizes Python Embedding, which is called using the PYTHON_NUMPY keyword in the observation input template settings. The same Python script can processes both forecast and observation datasets, but only the observation dataset is not set up for native ingest by MET. Two separate forecast fields are verified against two respective observation fields, with the Python script being passed the input file, the model name, the variable name being analyzed, the initialization and valid times, and a flag to indicate if the field passed is observation or forecast. This process is repeated with 2 instance names to GridStat, each with a different setting for regridding, neighborhood evaluation, thresholding, output line types, and output prefix names.

METplus Workflow

GridStat is the only MET tool called in this example. It processes the following run time:

Init: 2022-07-03 12Z
Forecast lead: 36 hour

Because instance names are used, GridStat will run 2 times for this 1 initalization time.

METplus Configuration

METplus first loads the default configuration file found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line: parm/use_cases/model_applications/clouds/GridStat_fcstGFS_obsSATCORPS_cloudTopPressAndTemp.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/clouds/GridStat_
↪fcstMPAS_obsSATCORPS_lowAndTotalCloudFrac.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

# ###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###


PROCESS_LIST = GridStat, GridStat(nbr)
```

(continues on next page)

(continued from previous page)

```
###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###


LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2022070312
INIT_END=2022070312
INIT_INCREMENT = 12H

LEAD_SEQ = 36

LOOP_ORDER = times

###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/clouds/GridStat_fcstGFS_
↪obsSATCORPS_cloudTopPressAndTemp
FCST_GRID_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d}_gfs.t12z.pgrb2.0p25.f0{LEAD_SEQ}

OBS_GRID_STAT_INPUT_DIR =
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_CLIMO_MEAN_INPUT_DIR =
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/clouds/GridStat_fcstGFS_obsSATCORPS_
↪cloudTopPressAndTemp
GRID_STAT_OUTPUT_TEMPLATE =
```

(continues on next page)

(continued from previous page)

```
###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = GFS
OBTYPE = SATCORPS

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/clouds/GridStat_fcstGFS_obsSATCORPS_
↪cloudTopPressAndTemp

FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = R649
FCST_VAR1_THRESH = gt0, lt180.0, ge190.0, ge200.0, ge210.0, ge220.0, ge230.0, ge240.0, ge250.
↪0, ge260.0, ge270.0, ge280.0

OBS_VAR1_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstGFS_obsSATCORPS_cloudTopPressAndTemp/GEO-MRGD.{valid?fmt=%Y%j.%H}30.GRID.NC:
↪{OBTYPE}:cloudTopTemp:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:2
OBS_VAR1_LEVELS =
OBS_VAR1_THRESH = gt0, lt180.0, ge190.0, ge200.0, ge210.0, ge220.0, ge230.0, ge240.0, ge250.
↪0, ge260.0, ge270.0, ge280.0


FCST_VAR2_NAME = PRES
FCST_VAR2_LEVELS = R646
FCST_VAR2_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0
FCST_VAR2_OPTIONS = convert(x)=x*0.01;

OBS_VAR2_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstGFS_obsSATCORPS_cloudTopPressAndTemp/GEO-MRGD.{valid?fmt=%Y%j.%H}30.GRID.NC:
↪{OBTYPE}:cloudTopPres:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:2
OBS_VAR2_THRESH =  gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

#LOG_GRID_STAT_VERBOSITY = 2
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

GRID_STAT_REGRID_TO_GRID = FCST
GRID_STAT_REGRID_METHOD = BILIN
GRID_STAT_REGRID_WIDTH = 2


GRID_STAT_DESC =

FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0
OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_ONCE_PER_FIELD = False

FCST_IS_PROB = false

FCST_GRID_STAT_PROB_THRESH = ==0.1

OBS_IS_PROB = false

OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudHgts

GRID_STAT_OUTPUT_FLAG_FHO = STAT
GRID_STAT_OUTPUT_FLAG_CTC = BOTH
GRID_STAT_OUTPUT_FLAG_CTS = BOTH
GRID_STAT_OUTPUT_FLAG_CNT = STAT
GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_MASK_POLY = {INPUT_BASE}/model_applications/clouds/GridStat_fcstGFS_obsSATCORPS_
↪cloudTopPressAndTemp/GPP_17km_60S_60N_mask.nc

[nbr]

FCST_VAR1_THRESH = gt0, lt180.0, ge190.0, ge200.0, ge210.0, ge220.0, ge230.0, ge240.0, ge250.
↪0, ge260.0, ge270.0, ge280.0, >SFP20, >SFP30, >SFP40, >SFP50, >SFP60, >SFP70, >SFP80
OBS_VAR1_THRESH = gt0, lt180.0, ge190.0, ge200.0, ge210.0, ge220.0, ge230.0, ge240.0, ge250.
↪0, ge260.0, ge270.0, ge280.0, >SOP20, >SOP30, >SOP40, >SOP50, >SOP60, >SOP70, >SOP80

FCST_VAR2_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SFP20, >SFP30, >SFP40, >SFP50, >SFP60, >SFP70, >SFP80
OBS_VAR2_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SOP20, >SOP30, >SOP40, >SOP50, >SOP60, >SOP70, >SOP80


GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9
GRID_STAT_NEIGHBORHOOD_SHAPE = CIRCLE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >0.0

GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = NONE
GRID_STAT_OUTPUT_FLAG_CTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = NONE
GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTC = STAT
GRID_STAT_OUTPUT_FLAG_NBRCTS = STAT
GRID_STAT_OUTPUT_FLAG_NBRCNT = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = NONE

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_NBRHD = TRUE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudHgts_NBR
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
```

(continues on next page)

(continued from previous page)

```
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
```

(continues on next page)

(continued from previous page)

```
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case utilizes 1 Python script to read and process the observation fields. parm/use_cases/model_applications/clouds/GridStat_fcstGFS_obsSATCORPS_cloudTopPressAndTemp/read_input_da

```
#this code was provided by Craig Schwartz
#and is largely unaltered from its original
#function.

#from __future__ import print_function
import os
import sys
import numpy as np
import datetime as dt
from netCDF4 import Dataset  # http://code.google.com/p/netcdf4-python/
```

(continues on next page)

(continued from previous page)

```
from scipy.interpolate import NearestNDInterpolator, LinearNDInterpolator
#### for Plotting
import matplotlib.cm as cm
import matplotlib.axes as maxes
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
#from mpl_toolkits.basemap import Basemap
import fnmatch
import pygrib
import pickle as pk
#####

##########################################

missing_values = -9999.0  # for MET

# UPP top layer bounds (Pa) for cloud layers
PTOP_LOW_UPP  = 64200. # low for > 64200 Pa
PTOP_MID_UPP  = 35000. # mid between 35000-64200 Pa
PTOP_HIGH_UPP = 15000. # high between 15000-35000 Pa

# Values for 4 x 4 contingency table
Na, Nb, Nc, Nd = 1, 2, 3, 4
Ne, Nf, Ng, Nh = 5, 6, 7, 8
Ni, Nj, Nk, Nl = 9, 10, 11, 12
Nm, Nn, No, Np = 13, 14, 15, 16

# Notes:
# 1) Entry for 'point' is for point-to-point comparison and is all dummy data (except for␣
↪gridType) that is overwritten by point2point
# 2) ERA5 on NCAR CISL RDA changed at some point.  Old is ERA5_2017 (not used anymore), new␣
↪is ERA5, which we'll use for 2020 data
griddedDatasets =  {
   'MERRA2'    : { 'gridType':'LatLon', 'latVar':'lat',      'latDef':[-90.0,0.50,361], 'lonVar
↪':'lon',        'lonDef':[-180.0,0.625,576],   'flipY':True, 'ftype':'nc'},
   'SATCORPS' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[-180.0,0.3125,1152], 'flipY':False, 'ftype':'nc' },
   'ERA5_2017': { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.7848769072,0.
↪281016829130516,640], 'lonVar':'longitude', 'lonDef':[0.0,0.28125,1280], 'flipY':False,
↪'ftype':'nc' },
   'ERA5'      : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'GFS'       : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'GALWEM'    : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
```

(continues on next page)

(continued from previous page)

```
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':True, 'ftype':'grib'},
   'GALWEM17' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.921875,0.156250,
↪1152], 'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536], 'flipY':False, 'ftype':
↪'grib'},
   'WWMCA'    : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'MPAS'     : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721],
↪'lonVar':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'nc'},
   'SAT_WWMCA_MEAN' : { 'gridType':'LatLon', 'latVar':'lat','latDef':[-90.0,0.25,721],
↪'lonVar':'lon', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'point'    : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.156250,1152],
↪'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536],   'flipY':False, 'ftype':'nc'},
}
   #TODO:Correct one, but MET can ingest a Gaussian grid only in Grib2 format (from Randy B.)
   #'ERA5'     : { 'gridType':'Gaussian', 'nx':1280, 'ny':640, 'lon_zero':0, 'latVar':
↪'latitude', 'lonVar':'longitude', 'flipY':False, },

#GALWEM, both 17-km and 0.25-degree
lowCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':3,
↪'typeOfFirstFixedSurface':10, 'shortName':'lcc' }
midCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':4,
↪'typeOfFirstFixedSurface':10, 'shortName':'mcc' }
highCloudFrac_GALWEM =  { 'parameterCategory':6, 'parameterNumber':5,
↪'typeOfFirstFixedSurface':10, 'shortName':'hcc' }
totalCloudFrac_GALWEM = { 'parameterCategory':6, 'parameterNumber':1,
↪'typeOfFirstFixedSurface':10, 'shortName':'tcc' }
cloudTopHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':12,
↪'typeOfFirstFixedSurface':3, 'shortName':'cdct' }
cloudBaseHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':11,
↪'typeOfFirstFixedSurface':2, 'shortName':'cdcb' }

#GFS
lowCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':214, 'shortName':'tcc' }
midCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':224, 'shortName':'tcc' }
highCloudFrac_GFS =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':234, 'shortName':'tcc' }

#WWMCA
totalCloudFrac_WWMCA  = { 'parameterName':71, 'typeOfLevel':'entireAtmosphere', 'level':0 }

cloudTopHeightLev1_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':1 }
cloudTopHeightLev2_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':2 }
cloudTopHeightLev3_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':3 }
```

(continues on next page)

(continued from previous page)

```
cloudTopHeightLev4_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':4 }
cloudTopHeight_WWMCA      = [ cloudTopHeightLev1_WWMCA, cloudTopHeightLev2_WWMCA,␣
↪cloudTopHeightLev3_WWMCA, cloudTopHeightLev4_WWMCA ]

cloudBaseHeightLev1_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':1 }
cloudBaseHeightLev2_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':2 }
cloudBaseHeightLev3_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':3 }
cloudBaseHeightLev4_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':4 }
cloudBaseHeight_WWMCA      = [ cloudBaseHeightLev1_WWMCA, cloudBaseHeightLev2_WWMCA,␣
↪cloudBaseHeightLev3_WWMCA, cloudBaseHeightLev4_WWMCA ]

verifVariablesModel = {
    'binaryCloud'    :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
    'totalCloudFrac' :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
    'lowCloudFrac'   :  {'GFS':[lowCloudFrac_GFS], 'GALWEM17':[lowCloudFrac_GALWEM], 'GALWEM
↪':[lowCloudFrac_GALWEM], 'MPAS':['cldfrac_low_UM']},
    'midCloudFrac'   :  {'GFS':[midCloudFrac_GFS], 'GALWEM17':[midCloudFrac_GALWEM], 'GALWEM
↪':[midCloudFrac_GALWEM], 'MPAS':['cldfrac_mid_UM']},
    'highCloudFrac'  :  {'GFS':[highCloudFrac_GFS], 'GALWEM17':[highCloudFrac_GALWEM],
↪'GALWEM':[highCloudFrac_GALWEM], 'MPAS':['cldfrac_high_UM']},
    'cloudTopHeight' :  {'GFS':['']               , 'GALWEM17':[cloudTopHeight_GALWEM],
↪'GALWEM':[cloudTopHeight_GALWEM], 'MPAS':['cldht_top_UM']},
    'cloudBaseHeight' : {'GFS':['']               , 'GALWEM17':[cloudBaseHeight_GALWEM],
↪'GALWEM':[cloudBaseHeight_GALWEM], 'MPAS':['cldht_base_UM']},
}

cloudFracCatThresholds = '>0, <10.0, >=10.0, >=20.0, >=30.0, >=40.0, >=50.0, >=60.0, >=70.0,␣
↪>=80.0, >=90.0' # MET format string
brightnessTempThresholds = '<280.0, <275.0, <273.15, <270.0, <265.0, <260.0, <255.0, <250.0,
↪<245.0, <240.0, <235.0, <230.0, <225.0, <220.0, <215.0, <210.0, <=SFP1, <=SFP5, <=SFP10,
↪<=SFP25, <=SFP50, >=SFP50, >=SFP75, >=SFP90, >=SFP95, >=SFP99'
verifVariables = {
   'binaryCloud'    : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['TCC'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'NA',  'thresholds':'>0.0', 'interpMethod':'nearest' },
   'totalCloudFrac' : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['tcc'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin' },
   'lowCloudFrac'   : { 'MERRA2':['CLDLOW'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['lcc'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'midCloudFrac'   : { 'MERRA2':['CLDMID'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['MCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
```

(continues on next page)

(continued from previous page)

```
↪},
   'highCloudFrac'  : { 'MERRA2':['CLDHGH'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['HCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'cloudTopTemp'   : { 'MERRA2':['CLDTMP'], 'SATCORPS':['cloud_temperature_top_level'],
↪'ERA5':['']   , 'units':'K',   'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopPres'   : { 'MERRA2':['CLDPRS'], 'SATCORPS':['cloud_pressure_top_level'],
↪'ERA5':['']   , 'units':'hPa', 'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopHeight' : { 'MERRA2':['']      , 'SATCORPS':['cloud_height_top_level'],
↪'ERA5':['']   , 'WWMCA':cloudTopHeight_WWMCA,  'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
   'cloudBaseHeight': { 'MERRA2':['']      , 'SATCORPS':['cloud_height_base_level'],
↪'ERA5':['cbh'], 'WWMCA':cloudBaseHeight_WWMCA, 'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
   'cloudCeiling'   : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'m',   'thresholds':'NA', 'interpMethod':'bilin'},
   'brightnessTemp' : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'K',   'thresholds':brightnessTempThresholds, 'interpMethod':'bilin
↪'},
}

# Combine the two dictionaries
# Only reason verifVariablesModel exists is just for space--verifVaribles gets too long if␣
↪we keep adding more datasets
for key in verifVariablesModel.keys():
  x = verifVariablesModel[key]
  for key1 in x.keys():
     verifVariables[key][key1] = x[key1]

#f = '/glade/u/home/schwartz/cloud_verification/GFS_grib_0.25deg/2018112412/gfs.0p25.
↪2018112412.f006.grib2'
#grbs = pygrib.open(f)
#idx = pygrib.index(f,'parameterCategory','parameterNumber','typeOfFirstFixedSurface')
#model = 'GFS'
#variable = 'totCloudCover'
#x = verifVariablesModel[variable][model] # returns a list, whose ith element is a dictionary
# e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
#idx(parameterCategory=x[0]['parameterCategory'],parameterNumber=x[0]['parameterNumber'],
↪typeOfFirstFixedSurface=x[0]['typeOfFirstFixedSurface'])

# to read in an environmental variable
#x = os.getenv('a') # probably type string no matter what

###########
```

(continues on next page)

(continued from previous page)

```
def getThreshold(variable):
   x = verifVariables[variable]['thresholds']
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getInterpMethod(variable):
   x = verifVariables[variable]['interpMethod'].upper()
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getTotalCloudFrac(source,data):
   if source == 'SATCORPS':
    # x = data[0][0,:,:,0] * 1.0E-2  # scaling
      x = (data[0][0,:,:,1]  + data[0][0,:,:,2] + data[0][0,:,:,3])*1.0E-2  # scaling
   #  y = data[0]
   #  x = np.sum( y[:,:,:,1:4],axis=3)
   elif source == 'MERRA2':
#      x = ( data[0][0,:,:]+data[1][0,:,:]+data[2][0,:,:] ) *100.0 # the ith element of data␣
↪is a numpy array
      x = data[0][0,:,:] * 100.0 # the ith element of data is a numpy array
      print(x.min(), x.max())
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
      x = data[0][0,:,:] * 100.0
   elif source == 'SAT_WWMCA_MEAN':
      x = data[0][0,:,:] # already in %
   else:
      x = data[0]

   # This next line is WRONG.
   # Missing should be set to missing
   # Then, the non-missing values are 1s and 0s
   #output = np.where(x > 0.0, x, 0.0)
   #output = np.where(x < 0.0, -9999.0, x) # missing. currently used for SATCORPS

   x = np.where( x < 0.0  , 0.0,   x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%
   return x

def getBinaryCloud(source,data):
   y = getTotalCloudFrac(source,data)
   # keep NaNs as is, but then set everything else to either 100% or 0%
   x = np.where( np.isnan(y), y, np.where(y > 0.0, 100.0, 0.0) )
```

(continues on next page)

(continued from previous page)

```
   return x

def getLayerCloudFrac(source,data,layer):
   if source == 'SATCORPS':
      if layer.lower().strip() == 'low'  : i = 1
      if layer.lower().strip() == 'mid'  : i = 2
      if layer.lower().strip() == 'high' : i = 3
      x = data[0][0,:,:,i] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 100.0
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
      x = data[0][0,:,:] * 100.0
   else:
      x = data[0]

   x = np.where( x < 0.0, 0.0, x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%

   return x

def getCloudTopTemp(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
      x = data[0]
   return x

def getCloudTopPres(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E-1  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 1.0E-2  # scaling [Pa] -> [hPa]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
      x = data[0]
```

(continues on next page)

(continued from previous page)

```
   return x

def getCloudTopHeight(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
         print('error with WWMCA Cloud top height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmax(tmp,axis=0) # get maximum cloud top height across all layers
   else:
      x = data[0]

   # Eliminate unphysical values (assume cloud top shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y

def getCloudBaseHeight(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
```

(continues on next page)

(continued from previous page)

```
         print('error with WWMCA Cloud base height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmin(tmp,axis=0) # get lowest cloud base over all layers
   else:
      x = data[0]

   # Eliminate unphysical values (assume cloud base shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y

def getCloudCeiling(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0]   #TBD
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] # TBD
      except: x = data[0][0,:,:]
   return x

# add other functions for different variables

###########

def getDataArray(inputFile,source,variable,dataSource):
   # 1) inputFile:  File name--either observations or forecast
   # 2) source:     Obsevation source (e.g., MERRA, SATCORP, etc.)
   # 3) variable:   Variable to verify
   # 4) dataSource: If 1, process forecast file. If 2 process obs file.

#   # specifying names here temporarily. file names should be passed in to python from shell␣
↪script
#   if source == 'merra':      nc_file = '/gpfs/fs1/scratch/schwartz/MERRA/MERRA2_400.tavg1_
↪2d_rad_Nx.20181101.nc4'
#   elif source == 'satcorp':  nc_file = '/glade/scratch/bjung/met/test_satcorps/GEO-MRGD.
↪2018334.0000.GRID.NC'
#   elif source == 'era5':     nc_file = '/glade/scratch/bjung/met/test_era5/e5.oper.fc.sfc.
↪instan.128_164_tcc.regn320sc.2018111606_2018120112.nc'

   source = source.upper().strip()  # Force uppercase and get rid of blank spaces, for safety

   print('dataSource = ',dataSource)
```

(continues on next page)

(continued from previous page)

```
   ftype = griddedDatasets[source]['ftype'].lower().strip()

   # Get file handle
   if ftype == 'nc':
      nc_fid = Dataset(inputFile, "r", format="NETCDF4")
      #nc_fid.set_auto_scale(True)
   elif ftype == 'grib':
      if source == 'WWMCA':
        idx = pygrib.index(inputFile,'parameterName','typeOfLevel','level')
      else:
        idx = pygrib.index(inputFile,'parameterCategory','parameterNumber',
↪'typeOfFirstFixedSurface')

   # dataSource == 1 means forecast, 2 means obs
#  if dataSource == 1: varsToRead = verifVariablesModel[variable][source] # if ftype == 'grib
↪', returns a list whose ith element is a dictionary. otherwise, just a list
#  if dataSource == 2: varsToRead = verifVariables[variable][source] # returns a list
   varsToRead = verifVariables[variable][source] # if ftype == 'grib', returns a list whose␣
↪ith element is a dictionary. otherwise, just a list

   print('Trying to read ',inputFile)

   # Get lat/lon information--currently not used
  #latVar = griddedDatasets[source]['latVar']
  #lonVar = griddedDatasets[source]['lonVar']
  #lats = np.array(nc_fid.variables[latVar][:])   # extract/copy the data
  #lons = np.array(nc_fid.variables[lonVar][:] )

   #print(lats.max())
   #print(lons.max())

   # one way to deal with scale factors
   # probably using something like nc_fid.set_auto_scale(True) is better...
  #latMax = lats.max()
  #while latMax > 90.0:
  #   lons = lons * 0.1
  #   lats = lats * 0.1
  #   latMax = lats.max()

   # get data
   data = []
   for v in varsToRead:
      if ftype == 'grib':
         if source == 'WWMCA':
```

(continues on next page)

(continued from previous page)

```
          x = idx(parameterName=v['parameterName'],typeOfLevel=v['typeOfLevel'],level=v[
↪'level'])[0] # by getting element 0, you get a pygrib message
        else:
           # e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
           if ( variable == 'cloudTopHeight' or variable == 'cloudBaseHeight') and source␣
↪== 'GALWEM17':
              x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[1] # by getting␣
↪element 1, you get a pygrib message
           else:
              x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[0] # by getting␣
↪element 0, you get a pygrib message
           if x.shortName != v['shortName']: print('Name mismatch!')
           #ADDED BY JOHN O
           print(x)
           print('Reading ', x.shortName, 'at level ', x.typeOfFirstFixedSurface)
        read_var = x.values # same x.data()[0]
        read_missing = x.missingValue
        print('missing value = ',read_missing)

        # The missing value (read_missing) for GALWEM17 and GALWEM cloud base/height is␣
↪9999, which is not the best choice because
        # those could be actual values. So we need to use the masked array part (below) to␣
↪handle which
        # values are missing.  We also set read_missing to something unphysical to␣
↪essentially disable it.
        # Finally, if we don't change the 'missingValue' property in the GRIB2 file we are␣
↪eventually outputting,
        # the bitmap will get all messed up, because it will be based on 9999 instead of
↪$missing_values
        if variable == 'cloudTopHeight' or variable == 'cloudBaseHeight':
           read_missing = -9999.
           x['missingValue'] = read_missing
           if source == 'GALWEM17':
              #These are masked numpy arrays, with mask = True where there is a missing␣
↪value (no cloud)
              #Use np.ma.filled to create an ndarray where mask = True values are set to np.
↪nan
              read_var = np.ma.filled(read_var.astype(read_var.dtype), np.nan)
     elif ftype == 'nc':
        read_var = nc_fid.variables[v]         # extract/copy the data
        try:
           read_missing = read_var.missing_value  # get variable attributes. Each dataset␣
↪has own missing values.
```

(continues on next page)

(continued from previous page)

```
        except:
           read_missing = -9999. # set a default missing value. probably only need to do␣
↪this for MPAS

     print('Reading ', v)

     this_var = np.array( read_var )        # to numpy array
    #print(read_missing, np.nan)
     this_var = np.where( this_var==read_missing, np.nan, this_var )
    #print(this_var.shape)
     data.append(this_var) # ith element of the list is a NUMPY ARRAY for the ith variable
    #print(type(this_var))
    #print(type(data))

  # Call a function to get the variable of interest.
  # Add a new function for each variable
  if variable == 'binaryCloud':     raw_data = getBinaryCloud(source,data)
  if variable == 'totalCloudFrac':  raw_data = getTotalCloudFrac(source,data)
  if variable == 'lowCloudFrac':    raw_data = getLayerCloudFrac(source,data,'low')
  if variable == 'midCloudFrac':    raw_data = getLayerCloudFrac(source,data,'mid')
  if variable == 'highCloudFrac':   raw_data = getLayerCloudFrac(source,data,'high')
  if variable == 'cloudTopTemp':    raw_data = getCloudTopTemp(source,data)
  if variable == 'cloudTopPres':    raw_data = getCloudTopPres(source,data)
  if variable == 'cloudTopHeight':  raw_data = getCloudTopHeight(source,data)
  if variable == 'cloudBaseHeight': raw_data = getCloudBaseHeight(source,data)
  if variable == 'cloudCeiling':    raw_data = getCloudCeiling(source,data)

  raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

  # Array met_data is passed to MET
  # Graphics should plot $met_data to make sure things look correct
  if griddedDatasets[source]['flipY']:
     print('flipping ',source,' data about y-axis')
     met_data=np.flip(raw_data,axis=0).astype(float)
  else:
     met_data=raw_data.astype(float)

  # Make plotting optional or Just use plot_data_plane
#   plt_data=np.where(met_data<0, np.nan, met_data)
#   map=Basemap(projection='cyl',llcrnrlat=-90,urcrnrlat=90,llcrnrlon=-180,urcrnrlon=180,
↪resolution='c')
#   map.drawcoastlines()
#   map.drawcountries()
#   map.drawparallels(np.arange(-90,90,30),labels=[1,1,0,1])
```

(continues on next page)

(continued from previous page)

```
#   map.drawmeridians(np.arange(0,360,60),labels=[1,1,0,1])
#   plt.contourf(lons,lats,plt_data,20,origin='upper',cmap=cm.Greens) #cm.gist_rainbow)
#   title=source+"_"+variable+"_"+str(validTime)
#   plt.title(title)
#   plt.colorbar(orientation='horizontal')
#   plt.savefig(title+".png")

   # If a forecast file, output a GRIB file with
   # 1 record containing the met_data
   # This is a hack, because right now, MET python embedding doesn't work with pygrib,
   #    so output the data to a temporary file, and then have MET read the temporary grib␣
↪file.
   # Starting with version 9.0 of MET, the hack isn't needed, and MET python embedding works␣
↪with pygrib
   outputFcstFile = False  # MUST be True for MET version < 9.0.  For MET 9.0+, optional
   if dataSource == 1 and ftype == 'grib':
      if outputFcstFile:
         grbtmp = x
         grbtmp['values']=met_data
         grbout = open('temp_fcst.grb2','ab')
         grbout.write(grbtmp.tostring())
         grbout.close() # Close the outfile GRIB file
         print('Successfully output temp_fcst.grb2')

   # Close files
   if ftype == 'grib': idx.close()    # Close the input GRIB file
   if ftype == 'nc':   nc_fid.close() # Close the netCDF file

   return met_data

def obsError(fcstData,obsErrorFile,validDate,dataSource):

   print('Adding noise to the cloud fraction fields')
   print('Using obsErrorFile',obsErrorFile)

   # First load the obsError information
   #obsErrorFile = 'ob_errors.pk'
   infile = open(obsErrorFile,'rb')
   binEdges, binStddev = pk.load(infile) # 'numpy.ndarray' types
   infile.close()

   # Get 1d forecast data
   shape = fcstData.shape
   fcst = fcstData.flatten()
```

(continues on next page)

(continued from previous page)

```
   # Set random number seed based on valid time and model
   if   dataSource.upper().strip() == 'MPAS':   ii = 10
   elif dataSource.upper().strip() == 'GALWEM': ii = 20
   elif dataSource.upper().strip() == 'GFS':    ii = 30
   np.random.seed(int(validDate*.1 + ii))

   # Find which bin the data is in
   for i in range(0,len(binEdges)-1):
      idx = np.where( (fcst >= binEdges[i]) & (fcst < binEdges[i+1]) )[0]
      n = len(idx) # number of points in the ith bin
      if n > 0: # check for empty bins
         randVals = np.random.normal(0,binStddev[i],n)
         fcst[idx] = fcst[idx] + randVals

   # bound forecast values to between 0 and 100%
   fcst = np.where( fcst < 0.0,     0.0,   fcst)
   fcst = np.where( fcst > 100.0,   100.0, fcst)

   # now reshape forecast data back to 2D
   output = fcst.reshape(shape)

   # data will have NaNs where bad.
   return output

def getFcstCloudFrac(cfr,pmid,psfc,layerDefinitions): # cfr is cloud fraction(%), pmid is 3D␣
↪pressure(Pa), psfc is surface pressure (Pa) code from UPP ./INITPOST.F

   if pmid.shape != cfr.shape:  # sanity check
      print('dimension mismatch bewteen cldfra and pressure')
      sys.exit()

   nlocs, nlevs = pmid.shape

   if len(psfc) != nlocs: # another sanity check
      print('dimension mismatch bewteen cldfra and surface pressure')
      sys.exit()

   cfracl = np.zeros(nlocs)
   cfracm = np.zeros(nlocs)
   cfrach = np.zeros(nlocs)

   for i in range(0,nlocs):

      PTOP_HIGH = PTOP_HIGH_UPP
      if layerDefinitions.upper().strip() == 'ERA5':
```

(continues on next page)

(continued from previous page)

```
         PTOP_LOW = 0.8*psfc[i]
         PTOP_MID = 0.45*psfc[i]
      elif layerDefinitions.upper().strip() == 'UPP':
         PTOP_LOW = PTOP_LOW_UPP
         PTOP_MID = PTOP_MID_UPP

      idxLow  = np.where(   pmid[i,:] >= PTOP_LOW)[0] # using np.where with just 1 argument␣
↪returns tuple
      idxMid  = np.where(  (pmid[i,:] <  PTOP_LOW) & (pmid[i,:] >= PTOP_MID))[0]
      idxHigh = np.where(  (pmid[i,:] <  PTOP_MID) & (pmid[i,:] >= PTOP_HIGH))[0]

      # use conditions in case all indices are missing
      if (len(idxLow) >0 ):  cfracl[i] = np.max( cfr[i,idxLow] )
      if (len(idxMid) >0 ):  cfracm[i] = np.max( cfr[i,idxMid] )
      if (len(idxHigh) >0 ): cfrach[i] = np.max( cfr[i,idxHigh] )

   tmp = np.vstack( (cfracl,cfracm,cfrach)) # stack the rows into one 2d array
   cldfraMax = np.max(tmp,axis=0) # get maximum value across low/mid/high for each pixel␣
↪(minimum overlap assumption)

   # This is the fortran code put into python format...double loop unnecessary and slow
   #for i in range(0,nlocs):
   #   for k in range(0,nlevs):
   #      if pmid(i,k) >= PTOP_LOW:
   #         cfracl(i) = np.max( [cfracl(i),cfr(i,k)] ) # Low
   #      elif pmid(i,k) < PTOP_LOW and pmid(i,k) >= PTOP_MID:
   #         cfracm(i) = np.max( [cfracm(i),cfr(i,k)] ) # Mid
   #      elif pmid(i,k) < PTOP_MID and pmid(i,k) >= PTOP_HIGH: # High
   #         cfrach(i) = np.max( [cfrach(i),cfr(i,k)] )

   return cfracl, cfracm, cfrach, cldfraMax

def getGOES16LatLon(g16_data_file):

   # Start timer
   startTime = dt.datetime.utcnow()

   # designate dataset
   g16nc = Dataset(g16_data_file, 'r')

   # GOES-R projection info and retrieving relevant constants
   proj_info = g16nc.variables['goes_imager_projection']
   lon_origin = proj_info.longitude_of_projection_origin
   H = proj_info.perspective_point_height+proj_info.semi_major_axis
   r_eq = proj_info.semi_major_axis
```

(continues on next page)

(continued from previous page)

```
   r_pol = proj_info.semi_minor_axis

   # Data info
   lat_rad_1d = g16nc.variables['x'][:]
   lon_rad_1d = g16nc.variables['y'][:]

   # close file when finished
   g16nc.close()
   g16nc = None

   # create meshgrid filled with radian angles
   lat_rad,lon_rad = np.meshgrid(lat_rad_1d,lon_rad_1d)

   # lat/lon calc routine from satellite radian angle vectors

   lambda_0 = (lon_origin*np.pi)/180.0

   a_var = np.power(np.sin(lat_rad),2.0) + (np.power(np.cos(lat_rad),2.0)*(np.power(np.
↪cos(lon_rad),2.0)+(((r_eq*r_eq)/(r_pol*r_pol))*np.power(np.sin(lon_rad),2.0))))
   b_var = -2.0*H*np.cos(lat_rad)*np.cos(lon_rad)
   c_var = (H**2.0)-(r_eq**2.0)

   r_s = (-1.0*b_var - np.sqrt((b_var**2)-(4.0*a_var*c_var)))/(2.0*a_var)

   s_x = r_s*np.cos(lat_rad)*np.cos(lon_rad)
   s_y = - r_s*np.sin(lat_rad)
   s_z = r_s*np.cos(lat_rad)*np.sin(lon_rad)

   lat = (180.0/np.pi)*(np.arctan(((r_eq*r_eq)/(r_pol*r_pol))*((s_z/np.sqrt(((H-s_x)*(H-s_
↪x))+(s_y*s_y))))))
   lon = (lambda_0 - np.arctan(s_y/(H-s_x)))*(180.0/np.pi)

   # End timer
   endTime = dt.datetime.utcnow()
   time = (endTime - startTime).microseconds / (1000.0*1000.0)
   print('took %f4.1 seconds to get GOES16 lat/lon'%(time))

   return lon,lat # lat/lon are 2-d arrays

# --
def getGOESRetrivalData(goesFile,goesVar):

   if not os.path.exists(goesFile):
      print(goesFile+' not there. exit')
      sys.exit()
```

(continues on next page)

(continued from previous page)

```
   # First get GOES lat/lon
   goesLon2d, goesLat2d = getGOES16LatLon(goesFile) # 2-d arrays
   goesLon = goesLon2d.flatten() # 1-d arrays
   goesLat = goesLat2d.flatten()

   # Now open the file and get the data we want
   nc_goes = Dataset(goesFile, "r", format="NETCDF4")

   # If the next line is true (it should be), this indicates the variable needs to be treated
   #  as an "unsigned 16-bit integer". This is a pain.  So we must use the "astype" method
   #  to change the variable type BEFORE applying scale_factor and add_offset.  After the␣
↪conversion
   #  we then can manually apply the scale factor and offset
   #goesVar = 'PRES'
   goesVar = goesVar.strip() # for safety
   if nc_goes.variables[goesVar]._Unsigned.lower().strip() == 'true':
      nc_goes.set_auto_scale(False) # Don't automatically apply scale_factor and add_offset␣
↪to variable
      goesData2d = np.array( nc_goes.variables[goesVar]).astype(np.uint16)
      goesData2d = goesData2d * nc_goes.variables[goesVar].scale_factor + nc_goes.
↪variables[goesVar].add_offset
      goesQC2d  = np.array( nc_goes.variables['DQF']).astype(np.uint8)
   else:
      goesData2d = np.array( nc_goes.variables[goesVar])
      goesQC2d  = np.array( nc_goes.variables['DQF'])

   # Make variables 1-d
   goesQC  = goesQC2d.flatten()
   goesData = goesData2d.flatten()
   nc_goes.close()

   # Get rid of NaNs; base it on longitude
   goesData = goesData[~np.isnan(goesLon)] # Handle data arrays first before changing lat/
↪lon itself
   goesQC  = goesQC[~np.isnan(goesLon)]
   goesLon = goesLon[~np.isnan(goesLon)] # ~ is "logical not", also np.logical_not
   goesLat = goesLat[~np.isnan(goesLat)]
   if goesLon.shape != goesLat.shape:
      print('GOES lat/lon shape mismatch')
      sys.exit()

   # If goesQC == 0, good QC and there was a cloud with a valid pressure.
   # If goesQC == 4, no cloud; probably clear sky.
   # All other QC means no data, and we want to remove those points
```

(continues on next page)

(continued from previous page)

```
   idx = np.logical_or( goesQC == 0, goesQC == 4) # Only keep QC == 0 or 4
   goesData = goesData[idx]
   goesQC  = goesQC[idx]
   goesLon = goesLon[idx]
   goesLat = goesLat[idx]

   # Only QC with 0 or 4 are left; now set QC == 4 to missing to indicate clear sky
   goesData = np.where( goesQC != 0, missing_values, goesData)

   # Get longitude to between (0,360) for consistency with JEDI files (this check is applied␣
↪to JEDI files, too)
   goesLon = np.where( goesLon < 0, goesLon + 360.0, goesLon )

   print('Min GOES Lon = ',np.min(goesLon))
   print('Max GOES Lon = ',np.max(goesLon))

   return goesLon, goesLat, goesData

def point2point(source,inputDir,satellite,channel,goesFile,condition,layerDefinitions,
↪dataSource):

   # Static Variables for QC and obs
   qcVar  = 'brightness_temperature_'+str(channel)+'@EffectiveQC' #'@EffectiveQC0' # QC␣
↪variable
   obsVar = 'brightness_temperature_'+str(channel)+'@ObsValue'  # Observation variable

   # Get GOES-16 retrieval file with auxiliary information
   if 'abi' in satellite or 'ahi' in satellite:
      goesLon, goesLat, goesData = getGOESRetrivalData(goesFile,'PRES') # return 1-d arrays
      lonlatGOES = np.array( list(zip(goesLon, goesLat))) # lon/lat pairs for each GOES ob␣
↪(nobs_GOES, 2)
     #print('shape lonlatGOES = ',lonlatGOES.shape)
      print('getting data from ',goesFile)
      myGOESInterpolator = NearestNDInterpolator(lonlatGOES,goesData)

   # First check to see if there's a concatenated file with all obs.
   #  If so, use that.  If not, have to process one file per processor, which takes a lot␣
↪more time
   if os.path.exists(inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'):
      inputFiles =  [inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'] # needs to be in a list␣
↪since we loop over inputFiles
   else:
      # Get list of OMB files to process.  There is one file per processor.
      # Need to get them in order so they are called in the same order for the
      # forecast and observed passes through this subroutine.
```

(continues on next page)

(continued from previous page)

```
      files = os.listdir(inputDir)
      inputFiles = fnmatch.filter(files,'obsout*_'+satellite+'*nc4') # returns relative path␣
↪names
      inputFiles = [inputDir+'/'+s for s in inputFiles] # add on directory name
      inputFiles.sort() # Get in order from low to high
   if len(inputFiles) == 0: return -99999, -99999 # if no matching files, force a failure

   # Variable to pull for brightness temperature
#  if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@GsiHofXBc' # Forecast␣
↪variable
   if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@hofx' #'@depbg' # OMB
   if dataSource == 2: v = obsVar

   # Read the files and put data in array
   allData, allDataQC = [], []
   for inputFile in inputFiles:
      nc_fid = Dataset(inputFile, "r", format="NETCDF4") #Dataset is the class behavior to␣
↪open the file
      print('Trying to read ',v,' from ',inputFile)

      # Read forecast/obs data
      read_var = nc_fid.variables[v]         # extract/copy the data
   #  read_missing = read_var.missing_value  # get variable attributes. Each dataset has own␣
↪missing values.
      this_var = np.array( read_var )        # to numpy array
   #  this_var = np.where( this_var==read_missing, np.nan, this_var )

     #if dataSource == 1: # If true, we just read in OMB data, but we want B
     #   obsData = np.array( nc_fid.variables[obsVar])
     #   this_var = obsData - this_var # get background/forecast value (O - OMB = B)

      #Read QC data
      qcData = np.array(nc_fid.variables[qcVar])

      # Sanity check...shapes should match
      if qcData.shape != this_var.shape: return -99999, -99999

      if 'abi' in satellite or 'ahi' in satellite:

         # Get the GOES-16 retrieval data at the observation locations in this file
         #   GOES values < 0 mean clear sky
         lats = np.array(nc_fid.variables['latitude@MetaData'])
         lons = np.array(nc_fid.variables['longitude@MetaData'])

         # Get longitude to between (0,360) for consistency with GOES-16 files
```

(continues on next page)

(continued from previous page)

```
        lons = np.where( lons < 0, lons + 360.0, lons )

        lonlat = np.array( list(zip(lons,lats)))  # lon/lat pairs for each ob (nobs, 2)
        thisGOESData = myGOESInterpolator(lonlat) # GOES data at obs locations in this file.
↪ If pressure, units are hPa
        thisGOESData = thisGOESData * 100.0 # get into Pa

        #obsCldfra = np.array( nc_fid.variables['cloud_area_fraction@MetaData'] )*100.0 #␣
↪Get into %...observed cloud fraction (AHI/ABI only)

        geoValsFile = inputFile.replace('obsout','geoval')
        if not os.path.exists(geoValsFile):
           print(geoValsFile+' not there. exit')
           sys.exit()

        nc_fid2 = Dataset(geoValsFile, "r", format="NETCDF4")
        fcstCldfra = np.array( nc_fid2.variables['cloud_area_fraction_in_atmosphere_layer
↪'])*100.0 # Get into %
        pressure   = np.array( nc_fid2.variables['air_pressure']) # Pa
        pressure_edges   = np.array( nc_fid2.variables['air_pressure_levels']) # Pa
        psfc = pressure_edges[:,-1]  # Surface pressure (Pa)...array order is top down
        if layerDefinitions.upper().strip() == 'ERA5':
           PTOP_LOW = 0.8*psfc # these are arrays
           PTOP_MID = 0.45*psfc
           PTOP_HIGH = PTOP_HIGH_UPP * np.ones_like(psfc)
        elif layerDefinitions.upper().strip() == 'UPP':
           PTOP_LOW = PTOP_LOW_UPP # these are constants
           PTOP_MID = PTOP_MID_UPP
           PTOP_HIGH = PTOP_HIGH_UPP
        else:
           print('layerDefinitions = ',layerDefinitions,'is invalid. exit')
           sys.exit()
        fcstLow,fcstMid,fcstHigh,fcstTotCldFra = getFcstCloudFrac(fcstCldfra,pressure,psfc,
↪layerDefinitions) # get low/mid/high/total forecast cloud fractions for each ob
        nc_fid2.close()

        # Modify QC data based on correspondence between forecast and obs. qcData used to␣
↪select good data later
        # It's possible that there are multiple forecast layers, such that fcstLow,fcstMid,
↪fcstHigh are all > $cldfraThresh
        # However, GOES-16 CTP doesn't really account for layering.  So, we need to remove␣
↪layered clouds from the forecast,
        #   focusing only on the layers that we asked for when doing {low,mid,high}Only␣
↪conditions
        # The "|" is symbol for "np.logcal_or"
```

(continues on next page)

(continued from previous page)

```
        yes = 2.0
        no  = 0.0
        cldfraThresh = 20.0 # percent
        if qcData.shape == fcstTotCldFra.shape == thisGOESData.shape:  # these should all␣
↪match
            print('Using condition ',condition,'for ABI/AHI')

            # Note that "&" is "np.logical_and" for boolean (true/false) quantities.
            # Thus, each condition should be enclosed in parentheses
            if   condition.lower().strip() == 'clearOnly'.lower():  # clear in both forecast␣
↪and obs
                qcData = np.where( (fcstTotCldFra < cldfraThresh)  & (thisGOESData <= 0.0),␣
↪qcData, missing_values)
            elif condition.lower().strip() == 'cloudyOnly'.lower(): # cloudy in both␣
↪forecast and obs
                qcData = np.where( (fcstTotCldFra >= cldfraThresh) & (thisGOESData > 0.0),␣
↪qcData, missing_values)
            elif condition.lower().strip() == 'lowOnly'.lower(): # low clouds in both␣
↪forecast and obs
                fcstLow = np.where( (fcstMid >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstLow) # remove mid, high
                qcData = np.where( (fcstLow >= cldfraThresh) & ( thisGOESData >= PTOP_LOW),␣
↪qcData, missing_values)
            elif condition.lower().strip() == 'midOnly'.lower(): # mid clouds in both␣
↪forecast and obs
                fcstMid = np.where( (fcstLow >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstMid) # remove low, high
                qcData = np.where( (fcstMid >= cldfraThresh) & (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID),   qcData, missing_values)
            elif condition.lower().strip() == 'highOnly'.lower(): # high clouds in both␣
↪forecast and obs
                fcstHigh = np.where( (fcstLow >= cldfraThresh) | ( fcstMid >= cldfraThresh),␣
↪missing_values, fcstHigh) # remove mid, high
                qcData = np.where( (fcstHigh >= cldfraThresh) & (thisGOESData <  PTOP_MID) &␣
↪(thisGOESData >= PTOP_HIGH), qcData, missing_values)
            elif condition.lower().strip() == 'fcstLow'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
                qcData = np.where( fcstLow >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'fcstMid'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
                qcData = np.where( fcstMid >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'fcstHigh'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
                qcData = np.where( fcstHigh >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'cloudEventLow'.lower():
```

(continues on next page)

(continued from previous page)

```
            if dataSource == 1: this_var = np.where( fcstLow      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
            if dataSource == 2: this_var = np.where( thisGOESData >= PTOP_LOW, yes, no )
         elif condition.lower().strip() == 'cloudEventMid'.lower():
            if dataSource == 1: this_var = np.where( fcstMid      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
            if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID), yes, no )
         elif condition.lower().strip() == 'cloudEventHigh'.lower():
            if dataSource == 1: this_var = np.where( fcstHigh     >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
            if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_MID) &␣
↪(thisGOESData >= PTOP_HIGH), yes, no )
         elif condition.lower().strip() == 'cloudEventTot'.lower():
            if dataSource == 1: this_var = np.where( fcstTotCldFra >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
            if dataSource == 2: this_var = np.where( thisGOESData  > 0.0, yes, no )
         elif condition.lower().strip() == 'all':
            print("not doing any conditional verification or stratifying by event")
         else:
            print("condition = ",condition," not recognized.")
            sys.exit()
         #elif condition.lower().strip() == '4x4table'.lower():
           #if dataSource == 1:
           #   this_var = np.where( fcstLow >= cldfraThresh, yesLow, no )
           #   this_var = this_var + np.where( fcstMid >= cldfraThresh, yesMid, no )
           #   this_var = this_var + np.where( fcstHigh >= cldfraThresh, yesHigh, no )
         print('number removed = ', (qcData==missing_values).sum())
        #print('number passed   = ', qcData.shape[0] - (qcData==missing_values).sum())
      else:
         print('shape mismatch')
         return -99999, -99999

   # Append to arrays
   allData.append(this_var)
   allDataQC.append(qcData)

   nc_fid.close() # done with the file, so close it before going to next file in loop

# We're now all done looping over the individul files

# Get the indices with acceptable QC
allQC = np.concatenate(allDataQC) # Put list of numpy arrays into a single long 1-D numpy␣
↪array.  All QC data.
idx = np.where(allQC==0) # returns indices
```

(continues on next page)

(continued from previous page)

```
   # Now get all the forecast/observed brightness temperature data with acceptable QC
   this_var = np.concatenate(allData)[idx] # Put list of numpy arrays into a single long 1-D␣
↪numpy array. This is all the forecast/obs data with good QC
   numObs = this_var.shape[0] # number of points with good QC for this channel
   print('Number of obs :',numObs)

   # Assume all the points actually fit into a square grid. Get the side of the square (use␣
↪ceil to round up)
   if numObs > 0:
      l = np.ceil(np.sqrt(numObs)).astype('int') # Length of the side of the square

      # Make an array that can be reshaped into the square
      raw_data1D = np.full(l*l,np.nan) # Initialize 1D array of length l**2 to np.nan
      raw_data1D[0:numObs] = this_var[:] # Fill data to the extent possible. There will be␣
↪some np.nan values at the end
      raw_data = np.reshape(raw_data1D,(l,l)) # Reshape into "square grid"

      raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

      met_data=raw_data.astype(float) # Give MET this info

      # Now need to tell MET the "grid" for the data
      # Make a fake lat/lon grid going from 0.0 to 50.0 degrees, with the interval␣
↪determined by number of points
      griddedDatasets[source]['latDef'][0] = 0.0 # starting point
      griddedDatasets[source]['latDef'][1] = np.diff(np.linspace(0,50,l)).round(6)[0] #␣
↪interval (degrees)
      griddedDatasets[source]['latDef'][2] = int(l) # number of points
      griddedDatasets[source]['lonDef'][0:3] = griddedDatasets[source]['latDef']

      gridInfo = getGridInfo(source, griddedDatasets[source]['gridType']) # 'LatLon' gridType
      return met_data, gridInfo

   else:
      return -99999, -99999

###########
def getGridInfo(source,gridType):

   if gridType == 'LatLon':
      latDef = griddedDatasets[source]['latDef']
      lonDef = griddedDatasets[source]['lonDef']
      gridInfo = {
```

(continues on next page)

(continued from previous page)

```
        'type':      gridType,
        'name':      source,
        'lat_ll':    latDef[0], #-90.000,
        'lon_ll':    lonDef[0], #-180.000,
        'delta_lat': latDef[1], #0.5000,
        'delta_lon': lonDef[1], #0.625,
        'Nlat':      latDef[2], #361,
        'Nlon':      lonDef[2], #576,
     }
  elif gridType == 'Gaussian':
     gridInfo = {
       'type':     gridType,
       'name':     source,
       'nx':       griddedDatasets[source]['nx'],
       'ny':       griddedDatasets[source]['ny'],
       'lon_zero': griddedDatasets[source]['lon_zero'],
     }

  return gridInfo

def getAttrArray(source,variable,initTime,validTime):

  init = dt.datetime.strptime(initTime,"%Y%m%d%H")
  valid = dt.datetime.strptime(validTime,"%Y%m%d%H")
  lead, rem = divmod((valid-init).total_seconds(), 3600)

  attrs = {

     'valid': valid.strftime("%Y%m%d_%H%M%S"),
     'init':  init.strftime("%Y%m%d_%H%M%S"),
     'lead':  str(int(lead)),
     'accum': '000000',

     'name':      variable,  #'MERRA2_Cloud_Percentage'
     'long_name': variable,  #'Cloud Percentage Levels',
     'level':     'ALL',
     'units':     verifVariables[variable]['units'],

     'grid': getGridInfo(source,griddedDatasets[source]['gridType'])
  }

  #print(attrs)
  #print(griddedDatasets[source])

  return attrs
```

(continues on next page)

(continued from previous page)

```
######## END FUNCTIONS   ##########


#if __name__ == "__main__":
dataFile, dataSource, variable, i_date, v_date, flag = sys.argv[1].split(":")
met_data = getDataArray(dataFile,dataSource,variable,flag)
attrs = getAttrArray(dataSource,variable,i_date,v_date)
print(attrs)
```

Running METplus

Pass the use case configuration file to the run_metplus.py script along with any user-specific system configuration files if desired:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/clouds/GridStat_fcstGFS_
↪obsSATCORPS_cloudTopPressAndTemp.conf /path/to/user_system.conf
```

See *Running METplus* (page 26) for more information.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/clouds/GridStat_fcstGFS_obsSATCORPS_cloudTopPressAndTemp (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_GFS_to_SATCORPS_F36_CloudHgts_360000L_20220705_000000V_pairs.nc
- grid_stat_GFS_to_SATCORPS_F36_CloudHgts_360000L_20220705_000000V_ctc.txt
- grid_stat_GFS_to_SATCORPS_F36_CloudHgts_360000L_20220705_000000V_cts.txt
- grid_stat_GFS_to_SATCORPS_F36_CloudHgts_360000L_20220705_000000V.stat
- grid_stat_GFS_to_SATCORPS_F36_CloudHgts_NBR_360000L_20220705_000000V_pairs.nc
- grid_stat_GFS_to_SATCORPS_F36_CloudHgts_NBR_360000L_20220705_000000V.stat

Keywords

Note:

- GridStatToolUseCase
- NetCDFFileUseCase
- CloudsAppUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/clouds-GridStat_fcstGFS_obsSATCORPS_cloudTopPressAndTemp.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.3.4 GridStat: Cloud Fractions with Neighborhood and Probabilities

model_applications/clouds/GridStat_fcstGFS_obsMERRA2_lowAndTotalCloudFrac.conf

Scientific Objective

This use case captures various statistical measures of two model comparisons for low and total cloud fraction with different neighborhood and probability settings for internal model metrics and to aid in future model updates

Datasets

Forecast: Global Forecast System (GFS)
Observations: Modern-Era Retrospective analysis for Research and Applications, Version 2 (MERRA2)
Grid: GPP 17km masking region

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

METplus Components

This use case utilizes Python Embedding, which is called using the PYTHON_NUMPY keyword in the observation input template settings. The same Python script processes both forecast and observation datasets, but only the observation dataset is not set up for native ingest by MET. Two separate forecast fields are verified against two respective observation fields, with the Python script being passed the input file, the model name, the variable name being analyzed, the initialization and valid times, and a flag to indicate if the field passed is observation or forecast. This process is repeated with 3 instance names to GridStat, each with a different setting for regridding, neighborhood evaluation, thresholding, output line types, and output prefix names.

METplus Workflow

GridStat is the only MET tool called in this example. It processes the following run time:

Init: 2021-07-03 12Z
Forecast lead: 36 hour

Because instance names are used, GridStat will run 3 times for this 1 initalization time.

METplus Configuration

METplus first loads the default configuration file found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line: parm/use_cases/model_applications/clouds/GridStat_fcstGFS_obsMERRA2_lowAndTotalCloudFrac.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/clouds/GridStat_
↪fcstGFS_obsMERRA2_lowAndTotalCloudFrac.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

# ###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###


PROCESS_LIST = GridStat, GridStat(nbr), GridStat(prob)

###
```

(continues on next page)

(continued from previous page)

```
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2021070312
INIT_END=2021070312
INIT_INCREMENT = 12H

LEAD_SEQ = 36

LOOP_ORDER = times

###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/clouds/GridStat_fcstGFS_obsMERRA2_
↪lowAndTotalCloudFrac
FCST_GRID_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d}.gfs.t12z.pgrb2.0p25.f0{LEAD_SEQ}

OBS_GRID_STAT_INPUT_DIR =
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_CLIMO_MEAN_INPUT_DIR =
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/clouds/GridStat_fcstGFS_obsMERRA2_
↪lowAndTotalCloudFrac
GRID_STAT_OUTPUT_TEMPLATE =
```

(continues on next page)

(continued from previous page)

```
###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = GFS
OBTYPE = MERRA2

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/clouds/GridStat_fcstGFS_obsMERRA2_
↪lowAndTotalCloudFrac

FCST_VAR1_NAME = TCDC
FCST_VAR1_LEVELS = R636
FCST_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0

OBS_VAR1_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstGFS_obsMERRA2_lowAndTotalCloudFrac/MERRA2_401.tavg1_2d_rad_Nx.{valid?fmt=%Y%m
↪%d}.nc4:{OBTYPE}:totalCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:2
OBS_VAR1_LEVELS =
OBS_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0


FCST_VAR2_NAME = LCDC
FCST_VAR2_LEVELS = R630
FCST_VAR2_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0

OBS_VAR2_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstGFS_obsMERRA2_lowAndTotalCloudFrac/MERRA2_401.tavg1_2d_rad_Nx.{valid?fmt=%Y%m
↪%d}.nc4:{OBTYPE}:lowCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:2
OBS_VAR2_THRESH =  gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

#LOG_GRID_STAT_VERBOSITY = 2

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_REGRID_TO_GRID = FCST
GRID_STAT_REGRID_METHOD = BILIN
GRID_STAT_REGRID_WIDTH = 2

GRID_STAT_DESC =

FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0
OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_ONCE_PER_FIELD = False

FCST_IS_PROB = false

FCST_GRID_STAT_PROB_THRESH = ==0.1

OBS_IS_PROB = false

OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudFracs

GRID_STAT_OUTPUT_FLAG_FHO = STAT
GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_CNT = STAT
GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_MASK_POLY = {INPUT_BASE}/model_applications/clouds/GridStat_fcstGFS_obsMERRA2_
↪lowAndTotalCloudFrac/GPP_17km_60S_60N_mask.nc
```

(continues on next page)

(continued from previous page)

```
[nbr]

FCST_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SFP20, >SFP30, >SFP40, >SFP50, >SFP60, >SFP70, >SFP80
OBS_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SOP20, >SOP30, >SOP40, >SOP50, >SOP60, >SOP70, >SOP80

FCST_VAR2_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SFP20, >SFP30, >SFP40, >SFP50, >SFP60, >SFP70, >SFP80
OBS_VAR2_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SOP20, >SOP30, >SOP40, >SOP50, >SOP60, >SOP70, >SOP80


GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9
GRID_STAT_NEIGHBORHOOD_SHAPE = CIRCLE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >0.0

GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = NONE
GRID_STAT_OUTPUT_FLAG_CTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = NONE
GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTC = STAT
GRID_STAT_OUTPUT_FLAG_NBRCTS = STAT
GRID_STAT_OUTPUT_FLAG_NBRCNT = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = NONE

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_NBRHD = TRUE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudFracs_NBR

[prob]

FCST_IS_PROB = TRUE

FCST_VAR1_THRESH = >0.1, >0.2, >0.3, >0.4, >0.5, >0.6, >0.7, >0.8, >0.9, >1.0
OBS_VAR1_THRESH = gt0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.0, ge90.0

FCST_VAR2_THRESH = >0.1, >0.2, >0.3, >0.4, >0.5, >0.6, >0.7, >0.8, >0.9, >1.0
```

(continues on next page)

(continued from previous page)

```
OBS_VAR2_THRESH = gt0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.0, ge90.0

GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9
GRID_STAT_NEIGHBORHOOD_SHAPE = CIRCLE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >0.0

GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = NONE
GRID_STAT_OUTPUT_FLAG_CTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = NONE
GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTC = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTS = NONE
GRID_STAT_OUTPUT_FLAG_NBRCNT = NONE
GRID_STAT_OUTPUT_FLAG_GRAD = NONE
GRID_STAT_OUTPUT_FLAG_PCT = STAT
GRID_STAT_OUTPUT_FLAG_PSTD = STAT
GRID_STAT_OUTPUT_FLAG_PJC = STAT
GRID_STAT_OUTPUT_FLAG_PRC = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_NBRHD = FALSE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudFracs_PROB
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
```

(continues on next page)

(continued from previous page)

```
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
```

(continues on next page)

(continued from previous page)

```
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case utilizes 1 Python script to read and process both forecast and observation fields. parm/use_cases/model_applications/clouds/GridStat_fcstGFS_obsMERRA2_lowAndTotalCloudFrac/read_input_data.

```
#this code was provided by Craig Schwartz
#and is largely unaltered from its original
#function.

#from __future__ import print_function
import os
import sys
import numpy as np
import datetime as dt
from netCDF4 import Dataset  # http://code.google.com/p/netcdf4-python/
from scipy.interpolate import NearestNDInterpolator, LinearNDInterpolator
#### for Plotting
import matplotlib.cm as cm
import matplotlib.axes as maxes
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
#from mpl_toolkits.basemap import Basemap
import fnmatch
import pygrib
import pickle as pk
#####

##########################################
```

(continues on next page)

(continued from previous page)

```
missing_values = -9999.0  # for MET

# UPP top layer bounds (Pa) for cloud layers
PTOP_LOW_UPP  = 64200. # low for > 64200 Pa
PTOP_MID_UPP  = 35000. # mid between 35000-64200 Pa
PTOP_HIGH_UPP = 15000. # high between 15000-35000 Pa

# Values for 4 x 4 contingency table
Na, Nb, Nc, Nd = 1, 2, 3, 4
Ne, Nf, Ng, Nh = 5, 6, 7, 8
Ni, Nj, Nk, Nl = 9, 10, 11, 12
Nm, Nn, No, Np = 13, 14, 15, 16

# Notes:
# 1) Entry for 'point' is for point-to-point comparison and is all dummy data (except for␣
↪gridType) that is overwritten by point2point
# 2) ERA5 on NCAR CISL RDA changed at some point.  Old is ERA5_2017 (not used anymore), new␣
↪is ERA5, which we'll use for 2020 data
griddedDatasets =  {
   'MERRA2'    : { 'gridType':'LatLon', 'latVar':'lat',     'latDef':[-90.0,0.50,361], 'lonVar
↪':'lon',       'lonDef':[-180.0,0.625,576],   'flipY':True, 'ftype':'nc'},
   'SATCORPS' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[-180.0,0.3125,1152], 'flipY':False, 'ftype':'nc' },
   'ERA5_2017': { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.7848769072,0.
↪281016829130516,640], 'lonVar':'longitude', 'lonDef':[0.0,0.28125,1280], 'flipY':False,
↪'ftype':'nc' },
   'ERA5'      : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'GFS'       : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'GALWEM'    : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':True, 'ftype':'grib'},
   'GALWEM17' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.921875,0.156250,
↪1152], 'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536], 'flipY':False, 'ftype':
↪'grib'},
   'WWMCA'     : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'MPAS'      : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721],
↪'lonVar':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'nc'},
   'SAT_WWMCA_MEAN' : { 'gridType':'LatLon', 'latVar':'lat','latDef':[-90.0,0.25,721],
↪'lonVar':'lon', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'point'     : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.156250,1152],
↪'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536],   'flipY':False, 'ftype':'nc'},
}
   #TODO:Correct one, but MET can ingest a Gaussian grid only in Grib2 format (from Randy B.)
```

(continues on next page)

(continued from previous page)

```
   #'ERA5'     : { 'gridType':'Gaussian', 'nx':1280, 'ny':640, 'lon_zero':0, 'latVar':
↪'latitude', 'lonVar':'longitude', 'flipY':False, },

#GALWEM, both 17-km and 0.25-degree
lowCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':3,
↪'typeOfFirstFixedSurface':10, 'shortName':'lcc' }
midCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':4,
↪'typeOfFirstFixedSurface':10, 'shortName':'mcc' }
highCloudFrac_GALWEM =  { 'parameterCategory':6, 'parameterNumber':5,
↪'typeOfFirstFixedSurface':10, 'shortName':'hcc' }
totalCloudFrac_GALWEM = { 'parameterCategory':6, 'parameterNumber':1,
↪'typeOfFirstFixedSurface':10, 'shortName':'tcc' }
cloudTopHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':12,
↪'typeOfFirstFixedSurface':3, 'shortName':'cdct' }
cloudBaseHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':11,
↪'typeOfFirstFixedSurface':2, 'shortName':'cdcb' }

#GFS
lowCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':214, 'shortName':'tcc' }
midCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':224, 'shortName':'tcc' }
highCloudFrac_GFS =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':234, 'shortName':'tcc' }

#WWMCA
totalCloudFrac_WWMCA  = { 'parameterName':71, 'typeOfLevel':'entireAtmosphere', 'level':0 }

cloudTopHeightLev1_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':1 }
cloudTopHeightLev2_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':2 }
cloudTopHeightLev3_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':3 }
cloudTopHeightLev4_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':4 }
cloudTopHeight_WWMCA      = [ cloudTopHeightLev1_WWMCA, cloudTopHeightLev2_WWMCA,␣
↪cloudTopHeightLev3_WWMCA, cloudTopHeightLev4_WWMCA ]

cloudBaseHeightLev1_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':1 }
cloudBaseHeightLev2_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':2 }
cloudBaseHeightLev3_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':3 }
cloudBaseHeightLev4_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':4 }
cloudBaseHeight_WWMCA      = [ cloudBaseHeightLev1_WWMCA, cloudBaseHeightLev2_WWMCA,␣
↪cloudBaseHeightLev3_WWMCA, cloudBaseHeightLev4_WWMCA ]

verifVariablesModel = {
    'binaryCloud'    :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
```

(continues on next page)

(continued from previous page)

```
    'totalCloudFrac' :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
    'lowCloudFrac'   :  {'GFS':[lowCloudFrac_GFS], 'GALWEM17':[lowCloudFrac_GALWEM], 'GALWEM
↪':[lowCloudFrac_GALWEM], 'MPAS':['cldfrac_low_UM']},
    'midCloudFrac'   :  {'GFS':[midCloudFrac_GFS], 'GALWEM17':[midCloudFrac_GALWEM], 'GALWEM
↪':[midCloudFrac_GALWEM], 'MPAS':['cldfrac_mid_UM']},
    'highCloudFrac'  :  {'GFS':[highCloudFrac_GFS], 'GALWEM17':[highCloudFrac_GALWEM],
↪'GALWEM':[highCloudFrac_GALWEM], 'MPAS':['cldfrac_high_UM']},
    'cloudTopHeight' :  {'GFS':['']               , 'GALWEM17':[cloudTopHeight_GALWEM],
↪'GALWEM':[cloudTopHeight_GALWEM], 'MPAS':['cldht_top_UM']},
    'cloudBaseHeight' : {'GFS':['']               , 'GALWEM17':[cloudBaseHeight_GALWEM],
↪'GALWEM':[cloudBaseHeight_GALWEM], 'MPAS':['cldht_base_UM']},
}

cloudFracCatThresholds = '>0, <10.0, >=10.0, >=20.0, >=30.0, >=40.0, >=50.0, >=60.0, >=70.0,␣
↪>=80.0, >=90.0' # MET format string
brightnessTempThresholds = '<280.0, <275.0, <273.15, <270.0, <265.0, <260.0, <255.0, <250.0,
↪<245.0, <240.0, <235.0, <230.0, <225.0, <220.0, <215.0, <210.0, <=SFP1, <=SFP5, <=SFP10,
↪<=SFP25, <=SFP50, >=SFP50, >=SFP75, >=SFP90, >=SFP95, >=SFP99'
verifVariables = {
   'binaryCloud'    : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['TCC'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'NA',  'thresholds':'>0.0', 'interpMethod':'nearest' },
   'totalCloudFrac' : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['tcc'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin' },
   'lowCloudFrac'   : { 'MERRA2':['CLDLOW'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['lcc'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'midCloudFrac'   : { 'MERRA2':['CLDMID'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['MCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'highCloudFrac'  : { 'MERRA2':['CLDHGH'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['HCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'cloudTopTemp'   : { 'MERRA2':['CLDTMP'], 'SATCORPS':['cloud_temperature_top_level'],
↪'ERA5':['']   , 'units':'K',   'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopPres'   : { 'MERRA2':['CLDPRS'], 'SATCORPS':['cloud_pressure_top_level'],
↪'ERA5':['']   , 'units':'hPa', 'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopHeight' : { 'MERRA2':['']      , 'SATCORPS':['cloud_height_top_level'],
↪'ERA5':['']   , 'WWMCA':cloudTopHeight_WWMCA,  'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
   'cloudBaseHeight': { 'MERRA2':['']      , 'SATCORPS':['cloud_height_base_level'],
↪'ERA5':['cbh'], 'WWMCA':cloudBaseHeight_WWMCA, 'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
```

(continues on next page)

(continued from previous page)

```
   'cloudCeiling'    : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'m',   'thresholds':'NA', 'interpMethod':'bilin'},
   'brightnessTemp' : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'K',   'thresholds':brightnessTempThresholds, 'interpMethod':'bilin
↪'},
}

# Combine the two dictionaries
# Only reason verifVariablesModel exists is just for space--verifVaribles gets too long if␣
↪we keep adding more datasets
for key in verifVariablesModel.keys():
  x = verifVariablesModel[key]
  for key1 in x.keys():
     verifVariables[key][key1] = x[key1]

#f = '/glade/u/home/schwartz/cloud_verification/GFS_grib_0.25deg/2018112412/gfs.0p25.
↪2018112412.f006.grib2'
#grbs = pygrib.open(f)
#idx = pygrib.index(f,'parameterCategory','parameterNumber','typeOfFirstFixedSurface')
#model = 'GFS'
#variable = 'totCloudCover'
#x = verifVariablesModel[variable][model] # returns a list, whose ith element is a dictionary
# e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
#idx(parameterCategory=x[0]['parameterCategory'],parameterNumber=x[0]['parameterNumber'],
↪typeOfFirstFixedSurface=x[0]['typeOfFirstFixedSurface'])

# to read in an environmental variable
#x = os.getenv('a') # probably type string no matter what

###########

def getThreshold(variable):
   x = verifVariables[variable]['thresholds']
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getInterpMethod(variable):
   x = verifVariables[variable]['interpMethod'].upper()
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getTotalCloudFrac(source,data):
   if source == 'SATCORPS':
    # x = data[0][0,:,:,0] * 1.0E-2  # scaling
      x = (data[0][0,:,:,1]  + data[0][0,:,:,2] + data[0][0,:,:,3])*1.0E-2  # scaling
```

(continues on next page)

(continued from previous page)

```
   #  y = data[0]
   #  x = np.sum( y[:,:,:,1:4],axis=3)
   elif source == 'MERRA2':
#      x = ( data[0][0,:,:]+data[1][0,:,:]+data[2][0,:,:] ) *100.0 # the ith element of data␣
↪is a numpy array
      x = data[0][0,:,:] * 100.0 # the ith element of data is a numpy array
      print(x.min(), x.max())
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
      x = data[0][0,:,:] * 100.0
   elif source == 'SAT_WWMCA_MEAN':
      x = data[0][0,:,:] # already in %
   else:
      x = data[0]

   # This next line is WRONG.
   # Missing should be set to missing
   # Then, the non-missing values are 1s and 0s
   #output = np.where(x > 0.0, x, 0.0)
   #output = np.where(x < 0.0, -9999.0, x) # missing. currently used for SATCORPS

   x = np.where( x < 0.0  , 0.0,   x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%
   return x

def getBinaryCloud(source,data):
   y = getTotalCloudFrac(source,data)
   # keep NaNs as is, but then set everything else to either 100% or 0%
   x = np.where( np.isnan(y), y, np.where(y > 0.0, 100.0, 0.0) )
   return x

def getLayerCloudFrac(source,data,layer):
   if source == 'SATCORPS':
      if layer.lower().strip() == 'low'  : i = 1
      if layer.lower().strip() == 'mid'  : i = 2
      if layer.lower().strip() == 'high' : i = 3
      x = data[0][0,:,:,i] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 100.0
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
```

(continues on next page)

(continued from previous page)

```
      x = data[0][0,:,:] * 100.0
   else:
      x = data[0]

   x = np.where( x < 0.0, 0.0, x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%

   return x

def getCloudTopTemp(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
      x = data[0]
   return x

def getCloudTopPres(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E-1  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 1.0E-2  # scaling [Pa] -> [hPa]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
      x = data[0]
   return x

def getCloudTopHeight(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
```

(continues on next page)

(continued from previous page)

```
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
         print('error with WWMCA Cloud top height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmax(tmp,axis=0) # get maximum cloud top height across all layers
   else:
      x = data[0]

   # Eliminate unphysical values (assume cloud top shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y

def getCloudBaseHeight(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
         print('error with WWMCA Cloud base height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmin(tmp,axis=0) # get lowest cloud base over all layers
   else:
      x = data[0]

   # Eliminate unphysical values (assume cloud base shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y

def getCloudCeiling(source,data):
```

(continues on next page)

(continued from previous page)

```
   if source == 'SATCORPS':
      x = data[0][0,:,:,0]   #TBD
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] # TBD
      except: x = data[0][0,:,:]
   return x

# add other functions for different variables

###########

def getDataArray(inputFile,source,variable,dataSource):
   # 1) inputFile:  File name--either observations or forecast
   # 2) source:     Obsevation source (e.g., MERRA, SATCORP, etc.)
   # 3) variable:   Variable to verify
   # 4) dataSource: If 1, process forecast file. If 2 process obs file.

#   # specifying names here temporarily. file names should be passed in to python from shell␣
↪script
#   if source == 'merra':      nc_file = '/gpfs/fs1/scratch/schwartz/MERRA/MERRA2_400.tavg1_
↪2d_rad_Nx.20181101.nc4'
#   elif source == 'satcorp':  nc_file = '/glade/scratch/bjung/met/test_satcorps/GEO-MRGD.
↪2018334.0000.GRID.NC'
#   elif source == 'era5':     nc_file = '/glade/scratch/bjung/met/test_era5/e5.oper.fc.sfc.
↪instan.128_164_tcc.regn320sc.2018111606_2018120112.nc'

   source = source.upper().strip()  # Force uppercase and get rid of blank spaces, for safety

   print('dataSource = ',dataSource)

   ftype = griddedDatasets[source]['ftype'].lower().strip()

   # Get file handle
   if ftype == 'nc':
      nc_fid = Dataset(inputFile, "r", format="NETCDF4")
      #nc_fid.set_auto_scale(True)
   elif ftype == 'grib':
      if source == 'WWMCA':
        idx = pygrib.index(inputFile,'parameterName','typeOfLevel','level')
      else:
        idx = pygrib.index(inputFile,'parameterCategory','parameterNumber',
↪'typeOfFirstFixedSurface')
```

(continues on next page)

(continued from previous page)

```
   # dataSource == 1 means forecast, 2 means obs
#  if dataSource == 1: varsToRead = verifVariablesModel[variable][source] # if ftype == 'grib
↪', returns a list whose ith element is a dictionary. otherwise, just a list
#  if dataSource == 2: varsToRead = verifVariables[variable][source] # returns a list
   varsToRead = verifVariables[variable][source] # if ftype == 'grib', returns a list whose␣
↪ith element is a dictionary. otherwise, just a list

   print('Trying to read ',inputFile)

   # Get lat/lon information--currently not used
  #latVar = griddedDatasets[source]['latVar']
  #lonVar = griddedDatasets[source]['lonVar']
  #lats = np.array(nc_fid.variables[latVar][:])   # extract/copy the data
  #lons = np.array(nc_fid.variables[lonVar][:] )

   #print(lats.max())
   #print(lons.max())

   # one way to deal with scale factors
   # probably using something like nc_fid.set_auto_scale(True) is better...
  #latMax = lats.max()
  #while latMax > 90.0:
  #   lons = lons * 0.1
  #   lats = lats * 0.1
  #   latMax = lats.max()

   # get data
   data = []
   for v in varsToRead:
      if ftype == 'grib':
         if source == 'WWMCA':
           x = idx(parameterName=v['parameterName'],typeOfLevel=v['typeOfLevel'],level=v[
↪'level'])[0] # by getting element 0, you get a pygrib message
         else:
            # e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
            if ( variable == 'cloudTopHeight' or variable == 'cloudBaseHeight') and source␣
↪== 'GALWEM17':
               x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[1] # by getting␣
↪element 1, you get a pygrib message
            else:
               x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[0] # by getting␣
↪element 0, you get a pygrib message
            if x.shortName != v['shortName']: print('Name mismatch!')
```

(continues on next page)

(continued from previous page)

```
            #ADDED BY JOHN O
            print(x)
            print('Reading ', x.shortName, 'at level ', x.typeOfFirstFixedSurface)
         read_var = x.values # same x.data()[0]
         read_missing = x.missingValue
         print('missing value = ',read_missing)

         # The missing value (read_missing) for GALWEM17 and GALWEM cloud base/height is␣
↪9999, which is not the best choice because
         # those could be actual values. So we need to use the masked array part (below) to␣
↪handle which
         # values are missing.  We also set read_missing to something unphysical to␣
↪essentially disable it.
         # Finally, if we don't change the 'missingValue' property in the GRIB2 file we are␣
↪eventually outputting,
         # the bitmap will get all messed up, because it will be based on 9999 instead of
↪$missing_values
         if variable == 'cloudTopHeight' or variable == 'cloudBaseHeight':
            read_missing = -9999.
            x['missingValue'] = read_missing
            if source == 'GALWEM17':
               #These are masked numpy arrays, with mask = True where there is a missing␣
↪value (no cloud)
               #Use np.ma.filled to create an ndarray where mask = True values are set to np.
↪nan
               read_var = np.ma.filled(read_var.astype(read_var.dtype), np.nan)
      elif ftype == 'nc':
         read_var = nc_fid.variables[v]         # extract/copy the data
         try:
            read_missing = read_var.missing_value  # get variable attributes. Each dataset␣
↪has own missing values.
         except:
            read_missing = -9999. # set a default missing value. probably only need to do␣
↪this for MPAS

      print('Reading ', v)

      this_var = np.array( read_var )        # to numpy array
     #print(read_missing, np.nan)
      this_var = np.where( this_var==read_missing, np.nan, this_var )
     #print(this_var.shape)
      data.append(this_var) # ith element of the list is a NUMPY ARRAY for the ith variable
     #print(type(this_var))
     #print(type(data))
```

(continues on next page)

(continued from previous page)

```
   # Call a function to get the variable of interest.
   # Add a new function for each variable
   if variable == 'binaryCloud':     raw_data = getBinaryCloud(source,data)
   if variable == 'totalCloudFrac':  raw_data = getTotalCloudFrac(source,data)
   if variable == 'lowCloudFrac':    raw_data = getLayerCloudFrac(source,data,'low')
   if variable == 'midCloudFrac':    raw_data = getLayerCloudFrac(source,data,'mid')
   if variable == 'highCloudFrac':   raw_data = getLayerCloudFrac(source,data,'high')
   if variable == 'cloudTopTemp':    raw_data = getCloudTopTemp(source,data)
   if variable == 'cloudTopPres':    raw_data = getCloudTopPres(source,data)
   if variable == 'cloudTopHeight':  raw_data = getCloudTopHeight(source,data)
   if variable == 'cloudBaseHeight': raw_data = getCloudBaseHeight(source,data)
   if variable == 'cloudCeiling':    raw_data = getCloudCeiling(source,data)

   raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

   # Array met_data is passed to MET
   # Graphics should plot $met_data to make sure things look correct
   if griddedDatasets[source]['flipY']:
      print('flipping ',source,' data about y-axis')
      met_data=np.flip(raw_data,axis=0).astype(float)
   else:
      met_data=raw_data.astype(float)

   # Make plotting optional or Just use plot_data_plane
#   plt_data=np.where(met_data<0, np.nan, met_data)
#   map=Basemap(projection='cyl',llcrnrlat=-90,urcrnrlat=90,llcrnrlon=-180,urcrnrlon=180,
↪resolution='c')
#   map.drawcoastlines()
#   map.drawcountries()
#   map.drawparallels(np.arange(-90,90,30),labels=[1,1,0,1])
#   map.drawmeridians(np.arange(0,360,60),labels=[1,1,0,1])
#   plt.contourf(lons,lats,plt_data,20,origin='upper',cmap=cm.Greens) #cm.gist_rainbow)
#   title=source+"_"+variable+"_"+str(validTime)
#   plt.title(title)
#   plt.colorbar(orientation='horizontal')
#   plt.savefig(title+".png")

   # If a forecast file, output a GRIB file with
   # 1 record containing the met_data
   # This is a hack, because right now, MET python embedding doesn't work with pygrib,
   #    so output the data to a temporary file, and then have MET read the temporary grib␣
↪file.
   # Starting with version 9.0 of MET, the hack isn't needed, and MET python embedding works␣
↪with pygrib
```

(continues on next page)

(continued from previous page)

```
   outputFcstFile = False  # MUST be True for MET version < 9.0.  For MET 9.0+, optional
   if dataSource == 1 and ftype == 'grib':
      if outputFcstFile:
         grbtmp = x
         grbtmp['values']=met_data
         grbout = open('temp_fcst.grb2','ab')
         grbout.write(grbtmp.tostring())
         grbout.close() # Close the outfile GRIB file
         print('Successfully output temp_fcst.grb2')

   # Close files
   if ftype == 'grib': idx.close()    # Close the input GRIB file
   if ftype == 'nc':   nc_fid.close() # Close the netCDF file

   return met_data

def obsError(fcstData,obsErrorFile,validDate,dataSource):

   print('Adding noise to the cloud fraction fields')
   print('Using obsErrorFile',obsErrorFile)

   # First load the obsError information
   #obsErrorFile = 'ob_errors.pk'
   infile = open(obsErrorFile,'rb')
   binEdges, binStddev = pk.load(infile) # 'numpy.ndarray' types
   infile.close()

   # Get 1d forecast data
   shape = fcstData.shape
   fcst = fcstData.flatten()

   # Set random number seed based on valid time and model
   if   dataSource.upper().strip() == 'MPAS':   ii = 10
   elif dataSource.upper().strip() == 'GALWEM': ii = 20
   elif dataSource.upper().strip() == 'GFS':    ii = 30
   np.random.seed(int(validDate*.1 + ii))

   # Find which bin the data is in
   for i in range(0,len(binEdges)-1):
      idx = np.where( (fcst >= binEdges[i]) & (fcst < binEdges[i+1]) )[0]
      n = len(idx) # number of points in the ith bin
      if n > 0: # check for empty bins
         randVals = np.random.normal(0,binStddev[i],n)
         fcst[idx] = fcst[idx] + randVals
```

(continues on next page)

(continued from previous page)

```
   # bound forecast values to between 0 and 100%
   fcst = np.where( fcst < 0.0,     0.0,   fcst)
   fcst = np.where( fcst > 100.0,   100.0, fcst)

   # now reshape forecast data back to 2D
   output = fcst.reshape(shape)

   # data will have NaNs where bad.
   return output

def getFcstCloudFrac(cfr,pmid,psfc,layerDefinitions): # cfr is cloud fraction(%), pmid is 3D␣
↪pressure(Pa), psfc is surface pressure (Pa) code from UPP ./INITPOST.F

   if pmid.shape != cfr.shape:  # sanity check
      print('dimension mismatch bewteen cldfra and pressure')
      sys.exit()

   nlocs, nlevs = pmid.shape

   if len(psfc) != nlocs: # another sanity check
      print('dimension mismatch bewteen cldfra and surface pressure')
      sys.exit()

   cfracl = np.zeros(nlocs)
   cfracm = np.zeros(nlocs)
   cfrach = np.zeros(nlocs)

   for i in range(0,nlocs):

      PTOP_HIGH = PTOP_HIGH_UPP
      if layerDefinitions.upper().strip() == 'ERA5':
         PTOP_LOW = 0.8*psfc[i]
         PTOP_MID = 0.45*psfc[i]
      elif layerDefinitions.upper().strip() == 'UPP':
         PTOP_LOW = PTOP_LOW_UPP
         PTOP_MID = PTOP_MID_UPP

      idxLow  = np.where(   pmid[i,:] >= PTOP_LOW)[0] # using np.where with just 1 argument␣
↪returns tuple
      idxMid  = np.where(  (pmid[i,:] <  PTOP_LOW) & (pmid[i,:] >= PTOP_MID))[0]
      idxHigh = np.where(  (pmid[i,:] <  PTOP_MID) & (pmid[i,:] >= PTOP_HIGH))[0]

      # use conditions in case all indices are missing
      if (len(idxLow) >0 ):  cfracl[i] = np.max( cfr[i,idxLow] )
      if (len(idxMid) >0 ):  cfracm[i] = np.max( cfr[i,idxMid] )
```

(continues on next page)

(continued from previous page)

```
      if (len(idxHigh) >0 ): cfrach[i] = np.max( cfr[i,idxHigh] )

   tmp = np.vstack( (cfracl,cfracm,cfrach)) # stack the rows into one 2d array
   cldfraMax = np.max(tmp,axis=0) # get maximum value across low/mid/high for each pixel␣
↪(minimum overlap assumption)

   # This is the fortran code put into python format...double loop unnecessary and slow
   #for i in range(0,nlocs):
   #   for k in range(0,nlevs):
   #      if pmid(i,k) >= PTOP_LOW:
   #         cfracl(i) = np.max( [cfracl(i),cfr(i,k)] ) # Low
   #      elif pmid(i,k) < PTOP_LOW and pmid(i,k) >= PTOP_MID:
   #         cfracm(i) = np.max( [cfracm(i),cfr(i,k)] ) # Mid
   #      elif pmid(i,k) < PTOP_MID and pmid(i,k) >= PTOP_HIGH: # High
   #         cfrach(i) = np.max( [cfrach(i),cfr(i,k)] )

   return cfracl, cfracm, cfrach, cldfraMax

def getGOES16LatLon(g16_data_file):

   # Start timer
   startTime = dt.datetime.utcnow()

   # designate dataset
   g16nc = Dataset(g16_data_file, 'r')

   # GOES-R projection info and retrieving relevant constants
   proj_info = g16nc.variables['goes_imager_projection']
   lon_origin = proj_info.longitude_of_projection_origin
   H = proj_info.perspective_point_height+proj_info.semi_major_axis
   r_eq = proj_info.semi_major_axis
   r_pol = proj_info.semi_minor_axis

   # Data info
   lat_rad_1d = g16nc.variables['x'][:]
   lon_rad_1d = g16nc.variables['y'][:]

   # close file when finished
   g16nc.close()
   g16nc = None

   # create meshgrid filled with radian angles
   lat_rad,lon_rad = np.meshgrid(lat_rad_1d,lon_rad_1d)

   # lat/lon calc routine from satellite radian angle vectors
```

(continues on next page)

(continued from previous page)

```
   lambda_0 = (lon_origin*np.pi)/180.0

   a_var = np.power(np.sin(lat_rad),2.0) + (np.power(np.cos(lat_rad),2.0)*(np.power(np.
↪cos(lon_rad),2.0)+(((r_eq*r_eq)/(r_pol*r_pol))*np.power(np.sin(lon_rad),2.0))))
   b_var = -2.0*H*np.cos(lat_rad)*np.cos(lon_rad)
   c_var = (H**2.0)-(r_eq**2.0)

   r_s = (-1.0*b_var - np.sqrt((b_var**2)-(4.0*a_var*c_var)))/(2.0*a_var)

   s_x = r_s*np.cos(lat_rad)*np.cos(lon_rad)
   s_y = - r_s*np.sin(lat_rad)
   s_z = r_s*np.cos(lat_rad)*np.sin(lon_rad)

   lat = (180.0/np.pi)*(np.arctan(((r_eq*r_eq)/(r_pol*r_pol))*((s_z/np.sqrt(((H-s_x)*(H-s_
↪x))+(s_y*s_y))))))
   lon = (lambda_0 - np.arctan(s_y/(H-s_x)))*(180.0/np.pi)

   # End timer
   endTime = dt.datetime.utcnow()
   time = (endTime - startTime).microseconds / (1000.0*1000.0)
   print('took %f4.1 seconds to get GOES16 lat/lon'%(time))

   return lon,lat # lat/lon are 2-d arrays

# --
def getGOESRetrivalData(goesFile,goesVar):

   if not os.path.exists(goesFile):
      print(goesFile+' not there. exit')
      sys.exit()

   # First get GOES lat/lon
   goesLon2d, goesLat2d = getGOES16LatLon(goesFile) # 2-d arrays
   goesLon = goesLon2d.flatten() # 1-d arrays
   goesLat = goesLat2d.flatten()

   # Now open the file and get the data we want
   nc_goes = Dataset(goesFile, "r", format="NETCDF4")

   # If the next line is true (it should be), this indicates the variable needs to be treated
   #  as an "unsigned 16-bit integer". This is a pain.  So we must use the "astype" method
   #  to change the variable type BEFORE applying scale_factor and add_offset.  After the␣
↪conversion
   #  we then can manually apply the scale factor and offset
```

(continues on next page)

(continued from previous page)

```
   #goesVar = 'PRES'
   goesVar = goesVar.strip() # for safety
   if nc_goes.variables[goesVar]._Unsigned.lower().strip() == 'true':
      nc_goes.set_auto_scale(False) # Don't automatically apply scale_factor and add_offset␣
↪to variable
      goesData2d = np.array( nc_goes.variables[goesVar]).astype(np.uint16)
      goesData2d = goesData2d * nc_goes.variables[goesVar].scale_factor + nc_goes.
↪variables[goesVar].add_offset
      goesQC2d  = np.array( nc_goes.variables['DQF']).astype(np.uint8)
   else:
      goesData2d = np.array( nc_goes.variables[goesVar])
      goesQC2d  = np.array( nc_goes.variables['DQF'])

   # Make variables 1-d
   goesQC  = goesQC2d.flatten()
   goesData = goesData2d.flatten()
   nc_goes.close()

   # Get rid of NaNs; base it on longitude
   goesData = goesData[~np.isnan(goesLon)] # Handle data arrays first before changing lat/
↪lon itself
   goesQC  = goesQC[~np.isnan(goesLon)]
   goesLon = goesLon[~np.isnan(goesLon)] # ~ is "logical not", also np.logical_not
   goesLat = goesLat[~np.isnan(goesLat)]
   if goesLon.shape != goesLat.shape:
      print('GOES lat/lon shape mismatch')
      sys.exit()

   # If goesQC == 0, good QC and there was a cloud with a valid pressure.
   # If goesQC == 4, no cloud; probably clear sky.
   # All other QC means no data, and we want to remove those points
   idx = np.logical_or( goesQC == 0, goesQC == 4) # Only keep QC == 0 or 4
   goesData = goesData[idx]
   goesQC  = goesQC[idx]
   goesLon = goesLon[idx]
   goesLat = goesLat[idx]

   # Only QC with 0 or 4 are left; now set QC == 4 to missing to indicate clear sky
   goesData = np.where( goesQC != 0, missing_values, goesData)

   # Get longitude to between (0,360) for consistency with JEDI files (this check is applied␣
↪to JEDI files, too)
   goesLon = np.where( goesLon < 0, goesLon + 360.0, goesLon )

   print('Min GOES Lon = ',np.min(goesLon))
```

(continues on next page)

(continued from previous page)

```
   print('Max GOES Lon = ',np.max(goesLon))

   return goesLon, goesLat, goesData

def point2point(source,inputDir,satellite,channel,goesFile,condition,layerDefinitions,
↪dataSource):

   # Static Variables for QC and obs
   qcVar  = 'brightness_temperature_'+str(channel)+'@EffectiveQC' #'@EffectiveQC0' # QC␣
↪variable
   obsVar = 'brightness_temperature_'+str(channel)+'@ObsValue'  # Observation variable

   # Get GOES-16 retrieval file with auxiliary information
   if 'abi' in satellite or 'ahi' in satellite:
      goesLon, goesLat, goesData = getGOESRetrivalData(goesFile,'PRES') # return 1-d arrays
      lonlatGOES = np.array( list(zip(goesLon, goesLat))) # lon/lat pairs for each GOES ob␣
↪(nobs_GOES, 2)
     #print('shape lonlatGOES = ',lonlatGOES.shape)
      print('getting data from ',goesFile)
      myGOESInterpolator = NearestNDInterpolator(lonlatGOES,goesData)

   # First check to see if there's a concatenated file with all obs.
   #  If so, use that.  If not, have to process one file per processor, which takes a lot␣
↪more time
   if os.path.exists(inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'):
      inputFiles =  [inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'] # needs to be in a list␣
↪since we loop over inputFiles
   else:
      # Get list of OMB files to process.  There is one file per processor.
      # Need to get them in order so they are called in the same order for the
      # forecast and observed passes through this subroutine.
      files = os.listdir(inputDir)
      inputFiles = fnmatch.filter(files,'obsout*_'+satellite+'*nc4') # returns relative path␣
↪names
      inputFiles = [inputDir+'/'+s for s in inputFiles] # add on directory name
      inputFiles.sort() # Get in order from low to high
   if len(inputFiles) == 0: return -99999, -99999 # if no matching files, force a failure

   # Variable to pull for brightness temperature
#  if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@GsiHofXBc' # Forecast␣
↪variable
   if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@hofx' #'@depbg' # OMB
   if dataSource == 2: v = obsVar

   # Read the files and put data in array
```

(continues on next page)

(continued from previous page)

```
   allData, allDataQC = [], []
   for inputFile in inputFiles:
      nc_fid = Dataset(inputFile, "r", format="NETCDF4") #Dataset is the class behavior to␣
↪open the file
      print('Trying to read ',v,' from ',inputFile)

      # Read forecast/obs data
      read_var = nc_fid.variables[v]         # extract/copy the data
   #  read_missing = read_var.missing_value  # get variable attributes. Each dataset has own␣
↪missing values.
      this_var = np.array( read_var )        # to numpy array
   #  this_var = np.where( this_var==read_missing, np.nan, this_var )

     #if dataSource == 1: # If true, we just read in OMB data, but we want B
     #   obsData = np.array( nc_fid.variables[obsVar])
     #   this_var = obsData - this_var # get background/forecast value (O - OMB = B)

      #Read QC data
      qcData = np.array(nc_fid.variables[qcVar])

      # Sanity check...shapes should match
      if qcData.shape != this_var.shape: return -99999, -99999

      if 'abi' in satellite or 'ahi' in satellite:

         # Get the GOES-16 retrieval data at the observation locations in this file
         #   GOES values < 0 mean clear sky
         lats = np.array(nc_fid.variables['latitude@MetaData'])
         lons = np.array(nc_fid.variables['longitude@MetaData'])

         # Get longitude to between (0,360) for consistency with GOES-16 files
         lons = np.where( lons < 0, lons + 360.0, lons )

         lonlat = np.array( list(zip(lons,lats)))  # lon/lat pairs for each ob (nobs, 2)
         thisGOESData = myGOESInterpolator(lonlat) # GOES data at obs locations in this file.
↪ If pressure, units are hPa
         thisGOESData = thisGOESData * 100.0 # get into Pa

         #obsCldfra = np.array( nc_fid.variables['cloud_area_fraction@MetaData'] )*100.0 #␣
↪Get into %...observed cloud fraction (AHI/ABI only)

         geoValsFile = inputFile.replace('obsout','geoval')
         if not os.path.exists(geoValsFile):
            print(geoValsFile+' not there. exit')
            sys.exit()
```

(continues on next page)

(continued from previous page)

```
        nc_fid2 = Dataset(geoValsFile, "r", format="NETCDF4")
        fcstCldfra = np.array( nc_fid2.variables['cloud_area_fraction_in_atmosphere_layer
↪'])*100.0 # Get into %
        pressure   = np.array( nc_fid2.variables['air_pressure']) # Pa
        pressure_edges   = np.array( nc_fid2.variables['air_pressure_levels']) # Pa
        psfc = pressure_edges[:,-1]  # Surface pressure (Pa)...array order is top down
        if layerDefinitions.upper().strip() == 'ERA5':
           PTOP_LOW = 0.8*psfc # these are arrays
           PTOP_MID = 0.45*psfc
           PTOP_HIGH = PTOP_HIGH_UPP * np.ones_like(psfc)
        elif layerDefinitions.upper().strip() == 'UPP':
           PTOP_LOW = PTOP_LOW_UPP # these are constants
           PTOP_MID = PTOP_MID_UPP
           PTOP_HIGH = PTOP_HIGH_UPP
        else:
           print('layerDefinitions = ',layerDefinitions,'is invalid. exit')
           sys.exit()
        fcstLow,fcstMid,fcstHigh,fcstTotCldFra = getFcstCloudFrac(fcstCldfra,pressure,psfc,
↪layerDefinitions) # get low/mid/high/total forecast cloud fractions for each ob
        nc_fid2.close()

        # Modify QC data based on correspondence between forecast and obs. qcData used to␣
↪select good data later
        # It's possible that there are multiple forecast layers, such that fcstLow,fcstMid,
↪fcstHigh are all > $cldfraThresh
        # However, GOES-16 CTP doesn't really account for layering.  So, we need to remove␣
↪layered clouds from the forecast,
        #   focusing only on the layers that we asked for when doing {low,mid,high}Only␣
↪conditions
        # The "|" is symbol for "np.logcal_or"
        yes = 2.0
        no  = 0.0
        cldfraThresh = 20.0 # percent
        if qcData.shape == fcstTotCldFra.shape == thisGOESData.shape:  # these should all␣
↪match
           print('Using condition ',condition,'for ABI/AHI')

           # Note that "&" is "np.logical_and" for boolean (true/false) quantities.
           # Thus, each condition should be enclosed in parentheses
           if   condition.lower().strip() == 'clearOnly'.lower():  # clear in both forecast␣
↪and obs
              qcData = np.where( (fcstTotCldFra < cldfraThresh)  & (thisGOESData <= 0.0),␣
↪qcData, missing_values)
           elif condition.lower().strip() == 'cloudyOnly'.lower(): # cloudy in both␣
```

(continues on next page)

(continued from previous page)

```
↪forecast and obs
                qcData = np.where( (fcstTotCldFra >= cldfraThresh) & (thisGOESData > 0.0),␣
↪qcData, missing_values)
            elif condition.lower().strip() == 'lowOnly'.lower(): # low clouds in both␣
↪forecast and obs
                fcstLow = np.where( (fcstMid >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstLow) # remove mid, high
                qcData = np.where( (fcstLow >= cldfraThresh) & ( thisGOESData >= PTOP_LOW),␣
↪qcData, missing_values)
            elif condition.lower().strip() == 'midOnly'.lower(): # mid clouds in both␣
↪forecast and obs
                fcstMid = np.where( (fcstLow >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstMid) # remove low, high
                qcData = np.where( (fcstMid >= cldfraThresh) & (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID),   qcData, missing_values)
            elif condition.lower().strip() == 'highOnly'.lower(): # high clouds in both␣
↪forecast and obs
                fcstHigh = np.where( (fcstLow >= cldfraThresh) | ( fcstMid >= cldfraThresh),␣
↪missing_values, fcstHigh) # remove mid, high
                qcData = np.where( (fcstHigh >= cldfraThresh) & (thisGOESData <  PTOP_MID) &␣
↪(thisGOESData >= PTOP_HIGH), qcData, missing_values)
            elif condition.lower().strip() == 'fcstLow'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
                qcData = np.where( fcstLow >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'fcstMid'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
                qcData = np.where( fcstMid >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'fcstHigh'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
                qcData = np.where( fcstHigh >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'cloudEventLow'.lower():
                if dataSource == 1: this_var = np.where( fcstLow      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
                if dataSource == 2: this_var = np.where( thisGOESData >= PTOP_LOW, yes, no )
            elif condition.lower().strip() == 'cloudEventMid'.lower():
                if dataSource == 1: this_var = np.where( fcstMid      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
                if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID), yes, no )
            elif condition.lower().strip() == 'cloudEventHigh'.lower():
                if dataSource == 1: this_var = np.where( fcstHigh     >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
                if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_MID) &␣
↪(thisGOESData >= PTOP_HIGH), yes, no )
            elif condition.lower().strip() == 'cloudEventTot'.lower():
```

(continues on next page)

(continued from previous page)

```
            if dataSource == 1: this_var = np.where( fcstTotCldFra >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
            if dataSource == 2: this_var = np.where( thisGOESData  > 0.0, yes, no )
         elif condition.lower().strip() == 'all':
            print("not doing any conditional verification or stratifying by event")
         else:
            print("condition = ",condition," not recognized.")
            sys.exit()
         #elif condition.lower().strip() == '4x4table'.lower():
           #if dataSource == 1:
           #   this_var = np.where( fcstLow >= cldfraThresh, yesLow, no )
           #   this_var = this_var + np.where( fcstMid >= cldfraThresh, yesMid, no )
           #   this_var = this_var + np.where( fcstHigh >= cldfraThresh, yesHigh, no )
         print('number removed = ', (qcData==missing_values).sum())
        #print('number passed   = ', qcData.shape[0] - (qcData==missing_values).sum())
      else:
         print('shape mismatch')
         return -99999, -99999

   # Append to arrays
   allData.append(this_var)
   allDataQC.append(qcData)

   nc_fid.close() # done with the file, so close it before going to next file in loop

# We're now all done looping over the individul files

# Get the indices with acceptable QC
allQC = np.concatenate(allDataQC) # Put list of numpy arrays into a single long 1-D numpy␣
↪array.  All QC data.
idx = np.where(allQC==0) # returns indices

# Now get all the forecast/observed brightness temperature data with acceptable QC
this_var = np.concatenate(allData)[idx] # Put list of numpy arrays into a single long 1-D␣
↪numpy array. This is all the forecast/obs data with good QC
numObs = this_var.shape[0] # number of points with good QC for this channel
print('Number of obs :',numObs)

# Assume all the points actually fit into a square grid. Get the side of the square (use␣
↪ceil to round up)
if numObs > 0:
   l = np.ceil(np.sqrt(numObs)).astype('int') # Length of the side of the square

   # Make an array that can be reshaped into the square
   raw_data1D = np.full(l*l,np.nan) # Initialize 1D array of length l**2 to np.nan
```

(continues on next page)

(continued from previous page)

```
      raw_data1D[0:numObs] = this_var[:] # Fill data to the extent possible. There will be␣
↪some np.nan values at the end
      raw_data = np.reshape(raw_data1D,(l,l)) # Reshape into "square grid"

      raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

      met_data=raw_data.astype(float) # Give MET this info

      # Now need to tell MET the "grid" for the data
      # Make a fake lat/lon grid going from 0.0 to 50.0 degrees, with the interval␣
↪determined by number of points
      griddedDatasets[source]['latDef'][0] = 0.0 # starting point
      griddedDatasets[source]['latDef'][1] = np.diff(np.linspace(0,50,l)).round(6)[0] #␣
↪interval (degrees)
      griddedDatasets[source]['latDef'][2] = int(l) # number of points
      griddedDatasets[source]['lonDef'][0:3] = griddedDatasets[source]['latDef']

      gridInfo = getGridInfo(source, griddedDatasets[source]['gridType']) # 'LatLon' gridType
      return met_data, gridInfo

   else:
      return -99999, -99999

############
def getGridInfo(source,gridType):

   if gridType == 'LatLon':
      latDef = griddedDatasets[source]['latDef']
      lonDef = griddedDatasets[source]['lonDef']
      gridInfo = {
         'type':      gridType,
         'name':      source,
         'lat_ll':    latDef[0], #-90.000,
         'lon_ll':    lonDef[0], #-180.000,
         'delta_lat': latDef[1], #0.5000,
         'delta_lon': lonDef[1], #0.625,
         'Nlat':      latDef[2], #361,
         'Nlon':      lonDef[2], #576,
      }
   elif gridType == 'Gaussian':
      gridInfo = {
        'type':     gridType,
        'name':     source,
        'nx':       griddedDatasets[source]['nx'],
```

(continues on next page)

(continued from previous page)

```
        'ny':       griddedDatasets[source]['ny'],
        'lon_zero': griddedDatasets[source]['lon_zero'],
      }

   return gridInfo

def getAttrArray(source,variable,initTime,validTime):

   init = dt.datetime.strptime(initTime,"%Y%m%d%H")
   valid = dt.datetime.strptime(validTime,"%Y%m%d%H")
   lead, rem = divmod((valid-init).total_seconds(), 3600)

   attrs = {

      'valid': valid.strftime("%Y%m%d_%H%M%S"),
      'init':  init.strftime("%Y%m%d_%H%M%S"),
      'lead':  str(int(lead)),
      'accum': '000000',

      'name':      variable,  #'MERRA2_Cloud_Percentage'
      'long_name': variable,  #'Cloud Percentage Levels',
      'level':     'ALL',
      'units':     verifVariables[variable]['units'],

      'grid': getGridInfo(source,griddedDatasets[source]['gridType'])
   }

   #print(attrs)
   #print(griddedDatasets[source])

   return attrs

######## END FUNCTIONS   ##########


#if __name__ == "__main__":
dataFile, dataSource, variable, i_date, v_date, flag = sys.argv[1].split(":")
met_data = getDataArray(dataFile,dataSource,variable,flag)
attrs = getAttrArray(dataSource,variable,i_date,v_date)
print(attrs)
```

Running METplus

Pass the use case configuration file to the run_metplus.py script along with any user-specific system configuration files if desired:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/clouds/GridStat_fcstGFS_
↪obsMERRA2_lowAndTotalCloudFrac.conf /path/to/user_system.conf
```

See *Running METplus* (page 26) for more information.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/clouds/GridStat_fcstGFS_obsMERRA2_lowAndTotalCloudFrac (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_GFS_TO_MERRA2_F36_CloudFracs_360000L_20210705_000000V_pairs.nc
- grid_stat_GFS_to_MERRA2_F36_CloudFracs_360000L_20210705_000000V.stat
- grid_stat_GFS_to_MERRA2_F36_CloudFracs_NBR_360000L_20210705_000000V_pairs.nc
- grid_stat_GFS_to_MERRA2_F36_CloudFracs_NBR_360000L_20210705_000000V.stat
- grid_stat_GFS_to_MERRA2_F36_CloudFracs_PROB_360000L_20210705_000000V_pairs.nc
- grid_stat_GFS_to_MERRA2_F36_CloudFracs_PROB_360000L_20210705_000000V.stat

Keywords

Note:

- GridStatToolUseCase
- NetCDFFileUseCase
- CloudsAppUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/clouds-GridStat_fcstGFS_obsMERRA2_lowAndTotalCloudFrac.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.3.5 GridStat: Cloud Fractions with Neighborhood and Probabilities

model_applications/clouds/GridStat_fcstMPAS_obsSATCORPS_lowAndTotalCloudFrac.conf

Scientific Objective

This use case captures various statistical measures of two model comparisons for low and total cloud fraction with different neighborhood and probability settings for internal model metrics and to aid in future model updates

Datasets

Forecast: Model for Prediction Across Scales (MPAS)
Observations: Satellite ClOud and Radiation Property retrieval System (SatCORPS)
Grid: GPP 17km masking region

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

METplus Components

This use case utilizes Python Embedding, which is called using the PYTHON_NUMPY keyword in the forecast and observation input template settings. The same Python script processes both forecast and observation datasets. Two separate forecast fields are verified against two respective observation fields, with the Python script being passed the input file, the model name, the variable name being analyzed, the initialization and valid times, and a flag to indicate if the field passed is observation or forecast. This process is repeated with 3 instance names to GridStat, each with a different setting for regridding, neighborhood evaluation, thresholding, output line types, and output prefix names.

METplus Workflow

GridStat is the only MET tool called in this example. It processes the following run time:

Init: 2020-07-23 00Z
Forecast lead: 36 hour

Because instance names are used, GridStat will run 3 times for this 1 initalization time.

METplus Configuration

METplus first loads the default configuration file found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line: parm/use_cases/model_applications/clouds/GridStat_fcstMPAS_obsSATCORPS_lowAndTotalCloudFrac.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/clouds/GridStat_
↪fcstMPAS_obsSATCORPS_lowAndTotalCloudFrac.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

# ###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###


PROCESS_LIST = GridStat, GridStat(nbr), GridStat(prob)

###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2020072300
INIT_END=2020072300
INIT_INCREMENT = 12H

LEAD_SEQ = 36

LOOP_ORDER = times

###
# File I/O
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR =
FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_GRID_STAT_INPUT_DIR =
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_CLIMO_MEAN_INPUT_DIR =
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/clouds/GridStat_fcstMPAS_obsSATCORPS_
↪lowAndTotalCloudFrac
GRID_STAT_OUTPUT_TEMPLATE =


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = MPAS
OBTYPE = SATCORPS

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/clouds/GridStat_fcstMPAS_obsSATCORPS_
↪lowAndTotalCloudFrac

FCST_VAR1_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstMPAS_obsSATCORPS_lowAndTotalCloudFrac/diag.{valid?fmt=%Y-%m-%d_%H}.00.00_
↪latlon.nc:{MODEL}:totalCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:1
FCST_VAR1_LEVELS =
FCST_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0

OBS_VAR1_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstMPAS_obsSATCORPS_lowAndTotalCloudFrac/GEO-MRGD.{valid?fmt=%Y%j.%H}00.GRID.NC:
↪{OBTYPE}:totalCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:2
OBS_VAR1_LEVELS =
OBS_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0
```

(continues on next page)

(continued from previous page)

```
FCST_VAR2_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstMPAS_obsSATCORPS_lowAndTotalCloudFrac/diag.{valid?fmt=%Y-%m-%d_%H}.00.00_
↪latlon.nc:{MODEL}:lowCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:1
FCST_VAR2_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0

OBS_VAR2_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstMPAS_obsSATCORPS_lowAndTotalCloudFrac/GEO-MRGD.{valid?fmt=%Y%j.%H}00.GRID.NC:
↪{OBTYPE}:lowCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:2
OBS_VAR2_THRESH =  gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

#LOG_GRID_STAT_VERBOSITY = 2

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped


GRID_STAT_REGRID_TO_GRID = FCST
GRID_STAT_REGRID_METHOD = BILIN
GRID_STAT_REGRID_WIDTH = 2


GRID_STAT_DESC =

FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0
OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_ONCE_PER_FIELD = False

FCST_IS_PROB = false
```

(continues on next page)

(continued from previous page)

```
FCST_GRID_STAT_PROB_THRESH = ==0.1

OBS_IS_PROB = false

OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = {MODEL}_F{lead?fmt=%H}_CloudFracs


GRID_STAT_OUTPUT_FLAG_FHO = STAT
GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_CNT = STAT
GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE


GRID_STAT_MASK_POLY = {INPUT_BASE}/model_applications/clouds/GridStat_fcstMPAS_obsSATCORPS_
↪lowAndTotalCloudFrac/GPP_17km_60S_60N_mask.nc

[nbr]

FCST_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SFP20, >SFP30, >SFP40, >SFP50, >SFP60, >SFP70, >SFP80

OBS_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SOP20, >SOP30, >SOP40, >SOP50, >SOP60, >SOP70, >SOP80

GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9
GRID_STAT_NEIGHBORHOOD_SHAPE = CIRCLE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >0.0

GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = NONE
GRID_STAT_OUTPUT_FLAG_CTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = NONE
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTC = STAT
GRID_STAT_OUTPUT_FLAG_NBRCTS = STAT
GRID_STAT_OUTPUT_FLAG_NBRCNT = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = NONE

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_NBRHD = TRUE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_OUTPUT_PREFIX = {MODEL}_F{lead?fmt=%H}_CloudFracs_NBR

[prob]

FCST_IS_PROB = TRUE

FCST_VAR1_THRESH = >0.1, >0.2, >0.3, >0.4, >0.5, >0.6, >0.7, >0.8, >0.9, >1.0
OBS_VAR1_THRESH = gt0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.0, ge90.0

FCST_VAR2_THRESH = >0.1, >0.2, >0.3, >0.4, >0.5, >0.6, >0.7, >0.8, >0.9, >1.0
OBS_VAR2_THRESH = gt0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.0, ge90.0

GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9
GRID_STAT_NEIGHBORHOOD_SHAPE = CIRCLE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >0.0

GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = NONE
GRID_STAT_OUTPUT_FLAG_CTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = NONE
GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTC = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTS = NONE
GRID_STAT_OUTPUT_FLAG_NBRCNT = NONE
GRID_STAT_OUTPUT_FLAG_GRAD = NONE
GRID_STAT_OUTPUT_FLAG_PCT = STAT
GRID_STAT_OUTPUT_FLAG_PSTD = STAT
GRID_STAT_OUTPUT_FLAG_PJC = STAT
GRID_STAT_OUTPUT_FLAG_PRC = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_NBRHD = FALSE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_OUTPUT_PREFIX = {MODEL}_F{lead?fmt=%H}_CloudFracs_PROB
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
```

(continues on next page)

(continued from previous page)

```
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
```

(continues on next page)

(continued from previous page)

```
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
```

(continues on next page)

(continued from previous page)

```
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case utilizes 1 Python script to read and process both forecast and observation fields. parm/use_cases/model_applications/clouds/GridStat_fcstMPAS_obsSATCORPS_lowAndTotalCloudFrac/read_input_d

```
#this code was provided by Craig Schwartz
#and is largely unaltered from its original
#function.

#from __future__ import print_function
import os
import sys
import numpy as np
import datetime as dt
from netCDF4 import Dataset  # http://code.google.com/p/netcdf4-python/
from scipy.interpolate import NearestNDInterpolator, LinearNDInterpolator
#### for Plotting
import matplotlib.cm as cm
import matplotlib.axes as maxes
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
#from mpl_toolkits.basemap import Basemap
import fnmatch
import pygrib
import pickle as pk
#####

###########################################

missing_values = -9999.0  # for MET

# UPP top layer bounds (Pa) for cloud layers
PTOP_LOW_UPP  = 64200. # low for > 64200 Pa
PTOP_MID_UPP  = 35000. # mid between 35000-64200 Pa
PTOP_HIGH_UPP = 15000. # high between 15000-35000 Pa

# Values for 4 x 4 contingency table
Na, Nb, Nc, Nd = 1, 2, 3, 4
Ne, Nf, Ng, Nh = 5, 6, 7, 8
Ni, Nj, Nk, Nl = 9, 10, 11, 12
Nm, Nn, No, Np = 13, 14, 15, 16

# Notes:
# 1) Entry for 'point' is for point-to-point comparison and is all dummy data (except for␣
↪gridType) that is overwritten by point2point
# 2) ERA5 on NCAR CISL RDA changed at some point.  Old is ERA5_2017 (not used anymore), new␣
↪is ERA5, which we'll use for 2020 data
```

(continues on next page)

(continued from previous page)

```
griddedDatasets =  {
   'MERRA2'    : { 'gridType':'LatLon', 'latVar':'lat',      'latDef':[-90.0,0.50,361], 'lonVar
↪':'lon',        'lonDef':[-180.0,0.625,576],   'flipY':True, 'ftype':'nc'},
   'SATCORPS' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[-180.0,0.3125,1152], 'flipY':False, 'ftype':'nc' },
   'ERA5_2017': { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.7848769072,0.
↪281016829130516,640], 'lonVar':'longitude', 'lonDef':[0.0,0.28125,1280], 'flipY':False,
↪'ftype':'nc' },
   'ERA5'      : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'GFS'       : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'GALWEM'    : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':True, 'ftype':'grib'},
   'GALWEM17' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.921875,0.156250,
↪1152], 'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536], 'flipY':False, 'ftype':
↪'grib'},
   'WWMCA'     : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'MPAS'      : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721],
↪'lonVar':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'nc'},
   'SAT_WWMCA_MEAN' : { 'gridType':'LatLon', 'latVar':'lat','latDef':[-90.0,0.25,721],
↪'lonVar':'lon', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'point'     : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.156250,1152],
↪'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536],   'flipY':False, 'ftype':'nc'},
}
   #TODO:Correct one, but MET can ingest a Gaussian grid only in Grib2 format (from Randy B.)
   #'ERA5'      : { 'gridType':'Gaussian', 'nx':1280, 'ny':640, 'lon_zero':0, 'latVar':
↪'latitude', 'lonVar':'longitude', 'flipY':False, },

#GALWEM, both 17-km and 0.25-degree
lowCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':3,
↪'typeOfFirstFixedSurface':10, 'shortName':'lcc' }
midCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':4,
↪'typeOfFirstFixedSurface':10, 'shortName':'mcc' }
highCloudFrac_GALWEM =  { 'parameterCategory':6, 'parameterNumber':5,
↪'typeOfFirstFixedSurface':10, 'shortName':'hcc' }
totalCloudFrac_GALWEM = { 'parameterCategory':6, 'parameterNumber':1,
↪'typeOfFirstFixedSurface':10, 'shortName':'tcc' }
cloudTopHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':12,
↪'typeOfFirstFixedSurface':3, 'shortName':'cdct' }
cloudBaseHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':11,
↪'typeOfFirstFixedSurface':2, 'shortName':'cdcb' }

#GFS
```

(continues on next page)

(continued from previous page)

```
lowCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':214, 'shortName':'tcc' }
midCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':224, 'shortName':'tcc' }
highCloudFrac_GFS =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':234, 'shortName':'tcc' }

#WWMCA
totalCloudFrac_WWMCA  = { 'parameterName':71, 'typeOfLevel':'entireAtmosphere', 'level':0 }

cloudTopHeightLev1_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':1 }
cloudTopHeightLev2_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':2 }
cloudTopHeightLev3_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':3 }
cloudTopHeightLev4_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':4 }
cloudTopHeight_WWMCA      = [ cloudTopHeightLev1_WWMCA, cloudTopHeightLev2_WWMCA,␣
↪cloudTopHeightLev3_WWMCA, cloudTopHeightLev4_WWMCA ]

cloudBaseHeightLev1_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':1 }
cloudBaseHeightLev2_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':2 }
cloudBaseHeightLev3_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':3 }
cloudBaseHeightLev4_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':4 }
cloudBaseHeight_WWMCA      = [ cloudBaseHeightLev1_WWMCA, cloudBaseHeightLev2_WWMCA,␣
↪cloudBaseHeightLev3_WWMCA, cloudBaseHeightLev4_WWMCA ]

verifVariablesModel = {
    'binaryCloud'    :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
    'totalCloudFrac' :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
    'lowCloudFrac'   :  {'GFS':[lowCloudFrac_GFS], 'GALWEM17':[lowCloudFrac_GALWEM], 'GALWEM
↪':[lowCloudFrac_GALWEM], 'MPAS':['cldfrac_low_UM']},
    'midCloudFrac'   :  {'GFS':[midCloudFrac_GFS], 'GALWEM17':[midCloudFrac_GALWEM], 'GALWEM
↪':[midCloudFrac_GALWEM], 'MPAS':['cldfrac_mid_UM']},
    'highCloudFrac'  :  {'GFS':[highCloudFrac_GFS], 'GALWEM17':[highCloudFrac_GALWEM],
↪'GALWEM':[highCloudFrac_GALWEM], 'MPAS':['cldfrac_high_UM']},
    'cloudTopHeight' :  {'GFS':['']               , 'GALWEM17':[cloudTopHeight_GALWEM],
↪'GALWEM':[cloudTopHeight_GALWEM], 'MPAS':['cldht_top_UM']},
    'cloudBaseHeight' : {'GFS':['']               , 'GALWEM17':[cloudBaseHeight_GALWEM],
↪'GALWEM':[cloudBaseHeight_GALWEM], 'MPAS':['cldht_base_UM']},
}

cloudFracCatThresholds = '>0, <10.0, >=10.0, >=20.0, >=30.0, >=40.0, >=50.0, >=60.0, >=70.0,␣
↪>=80.0, >=90.0' # MET format string
brightnessTempThresholds = '<280.0, <275.0, <273.15, <270.0, <265.0, <260.0, <255.0, <250.0,
↪<245.0, <240.0, <235.0, <230.0, <225.0, <220.0, <215.0, <210.0, <=SFP1, <=SFP5, <=SFP10,
```

(continues on next page)

(continued from previous page)

```
↪<=SFP25, <=SFP50, >=SFP50, >=SFP75, >=SFP90, >=SFP95, >=SFP99'
verifVariables = {
   'binaryCloud'    : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['TCC'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'NA',  'thresholds':'>0.0', 'interpMethod':'nearest' },
   'totalCloudFrac' : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['tcc'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin' },
   'lowCloudFrac'   : { 'MERRA2':['CLDLOW'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['lcc'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'midCloudFrac'   : { 'MERRA2':['CLDMID'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['MCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'highCloudFrac'  : { 'MERRA2':['CLDHGH'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['HCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'cloudTopTemp'   : { 'MERRA2':['CLDTMP'], 'SATCORPS':['cloud_temperature_top_level'],
↪'ERA5':['']   , 'units':'K',   'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopPres'   : { 'MERRA2':['CLDPRS'], 'SATCORPS':['cloud_pressure_top_level'],
↪'ERA5':['']   , 'units':'hPa', 'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopHeight' : { 'MERRA2':['']      , 'SATCORPS':['cloud_height_top_level'],
↪'ERA5':['']   , 'WWMCA':cloudTopHeight_WWMCA,  'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
   'cloudBaseHeight': { 'MERRA2':['']      , 'SATCORPS':['cloud_height_base_level'],
↪'ERA5':['cbh'], 'WWMCA':cloudBaseHeight_WWMCA, 'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
   'cloudCeiling'   : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'m',   'thresholds':'NA', 'interpMethod':'bilin'},
   'brightnessTemp' : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'K',   'thresholds':brightnessTempThresholds, 'interpMethod':'bilin
↪'},
}

# Combine the two dictionaries
# Only reason verifVariablesModel exists is just for space--verifVaribles gets too long if␣
↪we keep adding more datasets
for key in verifVariablesModel.keys():
  x = verifVariablesModel[key]
  for key1 in x.keys():
     verifVariables[key][key1] = x[key1]

#f = '/glade/u/home/schwartz/cloud_verification/GFS_grib_0.25deg/2018112412/gfs.0p25.
↪2018112412.f006.grib2'
#grbs = pygrib.open(f)
```

(continues on next page)

(continued from previous page)

```
#idx = pygrib.index(f,'parameterCategory','parameterNumber','typeOfFirstFixedSurface')
#model = 'GFS'
#variable = 'totCloudCover'
#x = verifVariablesModel[variable][model] # returns a list, whose ith element is a dictionary
# e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
#idx(parameterCategory=x[0]['parameterCategory'],parameterNumber=x[0]['parameterNumber'],
↪typeOfFirstFixedSurface=x[0]['typeOfFirstFixedSurface'])

# to read in an environmental variable
#x = os.getenv('a') # probably type string no matter what

###########

def getThreshold(variable):
   x = verifVariables[variable]['thresholds']
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getInterpMethod(variable):
   x = verifVariables[variable]['interpMethod'].upper()
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getTotalCloudFrac(source,data):
   if source == 'SATCORPS':
    # x = data[0][0,:,:,0] * 1.0E-2  # scaling
      x = (data[0][0,:,:,1]  + data[0][0,:,:,2] + data[0][0,:,:,3])*1.0E-2  # scaling
   #  y = data[0]
   #  x = np.sum( y[:,:,:,1:4],axis=3)
   elif source == 'MERRA2':
#      x = ( data[0][0,:,:]+data[1][0,:,:]+data[2][0,:,:] ) *100.0 # the ith element of data␣
↪is a numpy array
      x = data[0][0,:,:] * 100.0 # the ith element of data is a numpy array
      print(x.min(), x.max())
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
      x = data[0][0,:,:] * 100.0
   elif source == 'SAT_WWMCA_MEAN':
      x = data[0][0,:,:] # already in %
   else:
      x = data[0]

   # This next line is WRONG.
```

(continues on next page)

(continued from previous page)

```
   # Missing should be set to missing
   # Then, the non-missing values are 1s and 0s
   #output = np.where(x > 0.0, x, 0.0)
   #output = np.where(x < 0.0, -9999.0, x) # missing. currently used for SATCORPS

   x = np.where( x < 0.0  , 0.0,   x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%
   return x

def getBinaryCloud(source,data):
   y = getTotalCloudFrac(source,data)
   # keep NaNs as is, but then set everything else to either 100% or 0%
   x = np.where( np.isnan(y), y, np.where(y > 0.0, 100.0, 0.0) )
   return x

def getLayerCloudFrac(source,data,layer):
   if source == 'SATCORPS':
      if layer.lower().strip() == 'low'  : i = 1
      if layer.lower().strip() == 'mid'  : i = 2
      if layer.lower().strip() == 'high' : i = 3
      x = data[0][0,:,:,i] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 100.0
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
      x = data[0][0,:,:] * 100.0
   else:
      x = data[0]

   x = np.where( x < 0.0, 0.0, x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%

   return x

def getCloudTopTemp(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
```

(continues on next page)

(continued from previous page)

```
      x = data[0]
   return x

def getCloudTopPres(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E-1  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 1.0E-2  # scaling [Pa] -> [hPa]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
      x = data[0]
   return x

def getCloudTopHeight(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
         print('error with WWMCA Cloud top height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmax(tmp,axis=0) # get maximum cloud top height across all layers
   else:
      x = data[0]

   # Eliminate unphysical values (assume cloud top shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y

def getCloudBaseHeight(source,data):
   if source == 'SATCORPS':
```

(continues on next page)

(continued from previous page)

```
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
         print('error with WWMCA Cloud base height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmin(tmp,axis=0) # get lowest cloud base over all layers
   else:
      x = data[0]

   # Eliminate unphysical values (assume cloud base shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y

def getCloudCeiling(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0]   #TBD
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] # TBD
      except: x = data[0][0,:,:]
   return x

# add other functions for different variables

###########

def getDataArray(inputFile,source,variable,dataSource):
   # 1) inputFile:  File name--either observations or forecast
   # 2) source:     Obsevation source (e.g., MERRA, SATCORP, etc.)
   # 3) variable:   Variable to verify
   # 4) dataSource: If 1, process forecast file. If 2 process obs file.
```

(continues on next page)

(continued from previous page)

```
#   # specifying names here temporarily. file names should be passed in to python from shell␣
↪script
#   if source == 'merra':      nc_file = '/gpfs/fs1/scratch/schwartz/MERRA/MERRA2_400.tavg1_
↪2d_rad_Nx.20181101.nc4'
#   elif source == 'satcorp':  nc_file = '/glade/scratch/bjung/met/test_satcorps/GEO-MRGD.
↪2018334.0000.GRID.NC'
#   elif source == 'era5':     nc_file = '/glade/scratch/bjung/met/test_era5/e5.oper.fc.sfc.
↪instan.128_164_tcc.regn320sc.2018111606_2018120112.nc'

   source = source.upper().strip()  # Force uppercase and get rid of blank spaces, for safety

   print('dataSource = ',dataSource)

   ftype = griddedDatasets[source]['ftype'].lower().strip()

   # Get file handle
   if ftype == 'nc':
      nc_fid = Dataset(inputFile, "r", format="NETCDF4")
      #nc_fid.set_auto_scale(True)
   elif ftype == 'grib':
      if source == 'WWMCA':
        idx = pygrib.index(inputFile,'parameterName','typeOfLevel','level')
      else:
        idx = pygrib.index(inputFile,'parameterCategory','parameterNumber',
↪'typeOfFirstFixedSurface')

   # dataSource == 1 means forecast, 2 means obs
#  if dataSource == 1: varsToRead = verifVariablesModel[variable][source] # if ftype == 'grib
↪', returns a list whose ith element is a dictionary. otherwise, just a list
#  if dataSource == 2: varsToRead = verifVariables[variable][source] # returns a list
   varsToRead = verifVariables[variable][source] # if ftype == 'grib', returns a list whose␣
↪ith element is a dictionary. otherwise, just a list

   print('Trying to read ',inputFile)

   # Get lat/lon information--currently not used
  #latVar = griddedDatasets[source]['latVar']
  #lonVar = griddedDatasets[source]['lonVar']
  #lats = np.array(nc_fid.variables[latVar][:])   # extract/copy the data
  #lons = np.array(nc_fid.variables[lonVar][:] )

   #print(lats.max())
   #print(lons.max())
```

(continues on next page)

(continued from previous page)

```
  # one way to deal with scale factors
  # probably using something like nc_fid.set_auto_scale(True) is better...
 #latMax = lats.max()
 #while latMax > 90.0:
 #   lons = lons * 0.1
 #   lats = lats * 0.1
 #   latMax = lats.max()

  # get data
  data = []
  for v in varsToRead:
     if ftype == 'grib':
        if source == 'WWMCA':
          x = idx(parameterName=v['parameterName'],typeOfLevel=v['typeOfLevel'],level=v[
↪'level'])[0] # by getting element 0, you get a pygrib message
        else:
           # e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
           if ( variable == 'cloudTopHeight' or variable == 'cloudBaseHeight') and source␣
↪== 'GALWEM17':
              x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[1] # by getting␣
↪element 1, you get a pygrib message
           else:
              x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[0] # by getting␣
↪element 0, you get a pygrib message
           if x.shortName != v['shortName']: print('Name mismatch!')
           #ADDED BY JOHN O
           print(x)
           print('Reading ', x.shortName, 'at level ', x.typeOfFirstFixedSurface)
        read_var = x.values # same x.data()[0]
        read_missing = x.missingValue
        print('missing value = ',read_missing)

        # The missing value (read_missing) for GALWEM17 and GALWEM cloud base/height is␣
↪9999, which is not the best choice because
        # those could be actual values. So we need to use the masked array part (below) to␣
↪handle which
        # values are missing.  We also set read_missing to something unphysical to␣
↪essentially disable it.
        # Finally, if we don't change the 'missingValue' property in the GRIB2 file we are␣
↪eventually outputting,
        # the bitmap will get all messed up, because it will be based on 9999 instead of
↪$missing_values
        if variable == 'cloudTopHeight' or variable == 'cloudBaseHeight':
```

(continues on next page)

(continued from previous page)

```
            read_missing = -9999.
            x['missingValue'] = read_missing
            if source == 'GALWEM17':
               #These are masked numpy arrays, with mask = True where there is a missing␣
↪value (no cloud)
               #Use np.ma.filled to create an ndarray where mask = True values are set to np.
↪nan
               read_var = np.ma.filled(read_var.astype(read_var.dtype), np.nan)
      elif ftype == 'nc':
         read_var = nc_fid.variables[v]         # extract/copy the data
         try:
            read_missing = read_var.missing_value  # get variable attributes. Each dataset␣
↪has own missing values.
         except:
            read_missing = -9999. # set a default missing value. probably only need to do␣
↪this for MPAS

      print('Reading ', v)

      this_var = np.array( read_var )        # to numpy array
     #print(read_missing, np.nan)
      this_var = np.where( this_var==read_missing, np.nan, this_var )
     #print(this_var.shape)
      data.append(this_var) # ith element of the list is a NUMPY ARRAY for the ith variable
     #print(type(this_var))
     #print(type(data))

   # Call a function to get the variable of interest.
   # Add a new function for each variable
   if variable == 'binaryCloud':     raw_data = getBinaryCloud(source,data)
   if variable == 'totalCloudFrac':  raw_data = getTotalCloudFrac(source,data)
   if variable == 'lowCloudFrac':    raw_data = getLayerCloudFrac(source,data,'low')
   if variable == 'midCloudFrac':    raw_data = getLayerCloudFrac(source,data,'mid')
   if variable == 'highCloudFrac':   raw_data = getLayerCloudFrac(source,data,'high')
   if variable == 'cloudTopTemp':    raw_data = getCloudTopTemp(source,data)
   if variable == 'cloudTopPres':    raw_data = getCloudTopPres(source,data)
   if variable == 'cloudTopHeight':  raw_data = getCloudTopHeight(source,data)
   if variable == 'cloudBaseHeight': raw_data = getCloudBaseHeight(source,data)
   if variable == 'cloudCeiling':    raw_data = getCloudCeiling(source,data)

   raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

   # Array met_data is passed to MET
   # Graphics should plot $met_data to make sure things look correct
```

(continues on next page)

(continued from previous page)

```
   if griddedDatasets[source]['flipY']:
      print('flipping ',source,' data about y-axis')
      met_data=np.flip(raw_data,axis=0).astype(float)
   else:
      met_data=raw_data.astype(float)

   # Make plotting optional or Just use plot_data_plane
#   plt_data=np.where(met_data<0, np.nan, met_data)
#   map=Basemap(projection='cyl',llcrnrlat=-90,urcrnrlat=90,llcrnrlon=-180,urcrnrlon=180,
↪resolution='c')
#   map.drawcoastlines()
#   map.drawcountries()
#   map.drawparallels(np.arange(-90,90,30),labels=[1,1,0,1])
#   map.drawmeridians(np.arange(0,360,60),labels=[1,1,0,1])
#   plt.contourf(lons,lats,plt_data,20,origin='upper',cmap=cm.Greens) #cm.gist_rainbow)
#   title=source+"_"+variable+"_"+str(validTime)
#   plt.title(title)
#   plt.colorbar(orientation='horizontal')
#   plt.savefig(title+".png")

   # If a forecast file, output a GRIB file with
   # 1 record containing the met_data
   # This is a hack, because right now, MET python embedding doesn't work with pygrib,
   #    so output the data to a temporary file, and then have MET read the temporary grib␣
↪file.
   # Starting with version 9.0 of MET, the hack isn't needed, and MET python embedding works␣
↪with pygrib
   outputFcstFile = False  # MUST be True for MET version < 9.0.  For MET 9.0+, optional
   if dataSource == 1 and ftype == 'grib':
      if outputFcstFile:
         grbtmp = x
         grbtmp['values']=met_data
         grbout = open('temp_fcst.grb2','ab')
         grbout.write(grbtmp.tostring())
         grbout.close() # Close the outfile GRIB file
         print('Successfully output temp_fcst.grb2')

   # Close files
   if ftype == 'grib': idx.close()    # Close the input GRIB file
   if ftype == 'nc':   nc_fid.close() # Close the netCDF file

   return met_data

def obsError(fcstData,obsErrorFile,validDate,dataSource):
```

(continues on next page)

(continued from previous page)

```
   print('Adding noise to the cloud fraction fields')
   print('Using obsErrorFile',obsErrorFile)

   # First load the obsError information
   #obsErrorFile = 'ob_errors.pk'
   infile = open(obsErrorFile,'rb')
   binEdges, binStddev = pk.load(infile) # 'numpy.ndarray' types
   infile.close()

   # Get 1d forecast data
   shape = fcstData.shape
   fcst = fcstData.flatten()

   # Set random number seed based on valid time and model
   if   dataSource.upper().strip() == 'MPAS':   ii = 10
   elif dataSource.upper().strip() == 'GALWEM': ii = 20
   elif dataSource.upper().strip() == 'GFS':    ii = 30
   np.random.seed(int(validDate*.1 + ii))

   # Find which bin the data is in
   for i in range(0,len(binEdges)-1):
      idx = np.where( (fcst >= binEdges[i]) & (fcst < binEdges[i+1]) )[0]
      n = len(idx) # number of points in the ith bin
      if n > 0: # check for empty bins
         randVals = np.random.normal(0,binStddev[i],n)
         fcst[idx] = fcst[idx] + randVals

   # bound forecast values to between 0 and 100%
   fcst = np.where( fcst < 0.0,     0.0,   fcst)
   fcst = np.where( fcst > 100.0,   100.0, fcst)

   # now reshape forecast data back to 2D
   output = fcst.reshape(shape)

   # data will have NaNs where bad.
   return output

def getFcstCloudFrac(cfr,pmid,psfc,layerDefinitions): # cfr is cloud fraction(%), pmid is 3D␣
↪pressure(Pa), psfc is surface pressure (Pa) code from UPP ./INITPOST.F

   if pmid.shape != cfr.shape:  # sanity check
      print('dimension mismatch bewteen cldfra and pressure')
      sys.exit()

   nlocs, nlevs = pmid.shape
```

(continues on next page)

(continued from previous page)

```
   if len(psfc) != nlocs: # another sanity check
      print('dimension mismatch bewteen cldfra and surface pressure')
      sys.exit()

   cfracl = np.zeros(nlocs)
   cfracm = np.zeros(nlocs)
   cfrach = np.zeros(nlocs)

   for i in range(0,nlocs):

      PTOP_HIGH = PTOP_HIGH_UPP
      if layerDefinitions.upper().strip() == 'ERA5':
         PTOP_LOW = 0.8*psfc[i]
         PTOP_MID = 0.45*psfc[i]
      elif layerDefinitions.upper().strip() == 'UPP':
         PTOP_LOW = PTOP_LOW_UPP
         PTOP_MID = PTOP_MID_UPP

      idxLow  = np.where(   pmid[i,:] >= PTOP_LOW)[0] # using np.where with just 1 argument␣
↪returns tuple
      idxMid  = np.where(  (pmid[i,:] <  PTOP_LOW) & (pmid[i,:] >= PTOP_MID))[0]
      idxHigh = np.where(  (pmid[i,:] <  PTOP_MID) & (pmid[i,:] >= PTOP_HIGH))[0]

      # use conditions in case all indices are missing
      if (len(idxLow) >0 ):  cfracl[i] = np.max( cfr[i,idxLow] )
      if (len(idxMid) >0 ):  cfracm[i] = np.max( cfr[i,idxMid] )
      if (len(idxHigh) >0 ): cfrach[i] = np.max( cfr[i,idxHigh] )

   tmp = np.vstack( (cfracl,cfracm,cfrach)) # stack the rows into one 2d array
   cldfraMax = np.max(tmp,axis=0) # get maximum value across low/mid/high for each pixel␣
↪(minimum overlap assumption)

   # This is the fortran code put into python format...double loop unnecessary and slow
   #for i in range(0,nlocs):
   #   for k in range(0,nlevs):
   #      if pmid(i,k) >= PTOP_LOW:
   #         cfracl(i) = np.max( [cfracl(i),cfr(i,k)] ) # Low
   #      elif pmid(i,k) < PTOP_LOW and pmid(i,k) >= PTOP_MID:
   #         cfracm(i) = np.max( [cfracm(i),cfr(i,k)] ) # Mid
   #      elif pmid(i,k) < PTOP_MID and pmid(i,k) >= PTOP_HIGH: # High
   #         cfrach(i) = np.max( [cfrach(i),cfr(i,k)] )

   return cfracl, cfracm, cfrach, cldfraMax
```

(continues on next page)

(continued from previous page)

```
def getGOES16LatLon(g16_data_file):

   # Start timer
   startTime = dt.datetime.utcnow()

   # designate dataset
   g16nc = Dataset(g16_data_file, 'r')

   # GOES-R projection info and retrieving relevant constants
   proj_info = g16nc.variables['goes_imager_projection']
   lon_origin = proj_info.longitude_of_projection_origin
   H = proj_info.perspective_point_height+proj_info.semi_major_axis
   r_eq = proj_info.semi_major_axis
   r_pol = proj_info.semi_minor_axis

   # Data info
   lat_rad_1d = g16nc.variables['x'][:]
   lon_rad_1d = g16nc.variables['y'][:]

   # close file when finished
   g16nc.close()
   g16nc = None

   # create meshgrid filled with radian angles
   lat_rad,lon_rad = np.meshgrid(lat_rad_1d,lon_rad_1d)

   # lat/lon calc routine from satellite radian angle vectors

   lambda_0 = (lon_origin*np.pi)/180.0

   a_var = np.power(np.sin(lat_rad),2.0) + (np.power(np.cos(lat_rad),2.0)*(np.power(np.
↪cos(lon_rad),2.0)+(((r_eq*r_eq)/(r_pol*r_pol))*np.power(np.sin(lon_rad),2.0))))
   b_var = -2.0*H*np.cos(lat_rad)*np.cos(lon_rad)
   c_var = (H**2.0)-(r_eq**2.0)

   r_s = (-1.0*b_var - np.sqrt((b_var**2)-(4.0*a_var*c_var)))/(2.0*a_var)

   s_x = r_s*np.cos(lat_rad)*np.cos(lon_rad)
   s_y = - r_s*np.sin(lat_rad)
   s_z = r_s*np.cos(lat_rad)*np.sin(lon_rad)

   lat = (180.0/np.pi)*(np.arctan(((r_eq*r_eq)/(r_pol*r_pol))*((s_z/np.sqrt(((H-s_x)*(H-s_
↪x))+(s_y*s_y))))))
   lon = (lambda_0 - np.arctan(s_y/(H-s_x)))*(180.0/np.pi)
```

(continues on next page)

(continued from previous page)

```
   # End timer
   endTime = dt.datetime.utcnow()
   time = (endTime - startTime).microseconds / (1000.0*1000.0)
   print('took %f4.1 seconds to get GOES16 lat/lon'%(time))

   return lon,lat # lat/lon are 2-d arrays

# --
def getGOESRetrivalData(goesFile,goesVar):

   if not os.path.exists(goesFile):
      print(goesFile+' not there. exit')
      sys.exit()

   # First get GOES lat/lon
   goesLon2d, goesLat2d = getGOES16LatLon(goesFile) # 2-d arrays
   goesLon = goesLon2d.flatten() # 1-d arrays
   goesLat = goesLat2d.flatten()

   # Now open the file and get the data we want
   nc_goes = Dataset(goesFile, "r", format="NETCDF4")

   # If the next line is true (it should be), this indicates the variable needs to be treated
   #  as an "unsigned 16-bit integer". This is a pain.  So we must use the "astype" method
   #  to change the variable type BEFORE applying scale_factor and add_offset.  After the␣
↪conversion
   #  we then can manually apply the scale factor and offset
   #goesVar = 'PRES'
   goesVar = goesVar.strip() # for safety
   if nc_goes.variables[goesVar]._Unsigned.lower().strip() == 'true':
      nc_goes.set_auto_scale(False) # Don't automatically apply scale_factor and add_offset␣
↪to variable
      goesData2d = np.array( nc_goes.variables[goesVar]).astype(np.uint16)
      goesData2d = goesData2d * nc_goes.variables[goesVar].scale_factor + nc_goes.
↪variables[goesVar].add_offset
      goesQC2d  = np.array( nc_goes.variables['DQF']).astype(np.uint8)
   else:
      goesData2d = np.array( nc_goes.variables[goesVar])
      goesQC2d  = np.array( nc_goes.variables['DQF'])

   # Make variables 1-d
   goesQC  = goesQC2d.flatten()
   goesData = goesData2d.flatten()
   nc_goes.close()
```

(continues on next page)

(continued from previous page)

```
   # Get rid of NaNs; base it on longitude
   goesData = goesData[~np.isnan(goesLon)] # Handle data arrays first before changing lat/
↪lon itself
   goesQC  = goesQC[~np.isnan(goesLon)]
   goesLon = goesLon[~np.isnan(goesLon)] # ~ is "logical not", also np.logical_not
   goesLat = goesLat[~np.isnan(goesLat)]
   if goesLon.shape != goesLat.shape:
      print('GOES lat/lon shape mismatch')
      sys.exit()

   # If goesQC == 0, good QC and there was a cloud with a valid pressure.
   # If goesQC == 4, no cloud; probably clear sky.
   # All other QC means no data, and we want to remove those points
   idx = np.logical_or( goesQC == 0, goesQC == 4) # Only keep QC == 0 or 4
   goesData = goesData[idx]
   goesQC  = goesQC[idx]
   goesLon = goesLon[idx]
   goesLat = goesLat[idx]

   # Only QC with 0 or 4 are left; now set QC == 4 to missing to indicate clear sky
   goesData = np.where( goesQC != 0, missing_values, goesData)

   # Get longitude to between (0,360) for consistency with JEDI files (this check is applied␣
↪to JEDI files, too)
   goesLon = np.where( goesLon < 0, goesLon + 360.0, goesLon )

   print('Min GOES Lon = ',np.min(goesLon))
   print('Max GOES Lon = ',np.max(goesLon))

   return goesLon, goesLat, goesData

def point2point(source,inputDir,satellite,channel,goesFile,condition,layerDefinitions,
↪dataSource):

   # Static Variables for QC and obs
   qcVar  = 'brightness_temperature_'+str(channel)+'@EffectiveQC' #'@EffectiveQC0' # QC␣
↪variable
   obsVar = 'brightness_temperature_'+str(channel)+'@ObsValue'  # Observation variable

   # Get GOES-16 retrieval file with auxiliary information
   if 'abi' in satellite or 'ahi' in satellite:
      goesLon, goesLat, goesData = getGOESRetrivalData(goesFile,'PRES') # return 1-d arrays
      lonlatGOES = np.array( list(zip(goesLon, goesLat))) # lon/lat pairs for each GOES ob␣
↪(nobs_GOES, 2)
     #print('shape lonlatGOES = ',lonlatGOES.shape)
```

(continues on next page)

(continued from previous page)

```
      print('getting data from ',goesFile)
      myGOESInterpolator = NearestNDInterpolator(lonlatGOES,goesData)

   # First check to see if there's a concatenated file with all obs.
   #  If so, use that.  If not, have to process one file per processor, which takes a lot␣
↪more time
   if os.path.exists(inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'):
      inputFiles =  [inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'] # needs to be in a list␣
↪since we loop over inputFiles
   else:
      # Get list of OMB files to process.  There is one file per processor.
      # Need to get them in order so they are called in the same order for the
      # forecast and observed passes through this subroutine.
      files = os.listdir(inputDir)
      inputFiles = fnmatch.filter(files,'obsout*_'+satellite+'*nc4') # returns relative path␣
↪names
      inputFiles = [inputDir+'/'+s for s in inputFiles] # add on directory name
      inputFiles.sort() # Get in order from low to high
   if len(inputFiles) == 0: return -99999, -99999 # if no matching files, force a failure

   # Variable to pull for brightness temperature
#  if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@GsiHofXBc' # Forecast␣
↪variable
   if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@hofx' #'@depbg' # OMB
   if dataSource == 2: v = obsVar

   # Read the files and put data in array
   allData, allDataQC = [], []
   for inputFile in inputFiles:
      nc_fid = Dataset(inputFile, "r", format="NETCDF4") #Dataset is the class behavior to␣
↪open the file
      print('Trying to read ',v,' from ',inputFile)

      # Read forecast/obs data
      read_var = nc_fid.variables[v]         # extract/copy the data
   #  read_missing = read_var.missing_value  # get variable attributes. Each dataset has own␣
↪missing values.
      this_var = np.array( read_var )        # to numpy array
   #  this_var = np.where( this_var==read_missing, np.nan, this_var )

     #if dataSource == 1: # If true, we just read in OMB data, but we want B
     #   obsData = np.array( nc_fid.variables[obsVar])
     #   this_var = obsData - this_var # get background/forecast value (O - OMB = B)

      #Read QC data
```

(continues on next page)

(continued from previous page)

```
    qcData = np.array(nc_fid.variables[qcVar])

    # Sanity check...shapes should match
    if qcData.shape != this_var.shape: return -99999, -99999

    if 'abi' in satellite or 'ahi' in satellite:

       # Get the GOES-16 retrieval data at the observation locations in this file
       #   GOES values < 0 mean clear sky
       lats = np.array(nc_fid.variables['latitude@MetaData'])
       lons = np.array(nc_fid.variables['longitude@MetaData'])

       # Get longitude to between (0,360) for consistency with GOES-16 files
       lons = np.where( lons < 0, lons + 360.0, lons )

       lonlat = np.array( list(zip(lons,lats)))  # lon/lat pairs for each ob (nobs, 2)
       thisGOESData = myGOESInterpolator(lonlat) # GOES data at obs locations in this file.
↪ If pressure, units are hPa
       thisGOESData = thisGOESData * 100.0 # get into Pa

       #obsCldfra = np.array( nc_fid.variables['cloud_area_fraction@MetaData'] )*100.0 #␣
↪Get into %...observed cloud fraction (AHI/ABI only)

       geoValsFile = inputFile.replace('obsout','geoval')
       if not os.path.exists(geoValsFile):
          print(geoValsFile+' not there. exit')
          sys.exit()

       nc_fid2 = Dataset(geoValsFile, "r", format="NETCDF4")
       fcstCldfra = np.array( nc_fid2.variables['cloud_area_fraction_in_atmosphere_layer
↪'])*100.0 # Get into %
       pressure   = np.array( nc_fid2.variables['air_pressure']) # Pa
       pressure_edges   = np.array( nc_fid2.variables['air_pressure_levels']) # Pa
       psfc = pressure_edges[:,-1]  # Surface pressure (Pa)...array order is top down
       if layerDefinitions.upper().strip() == 'ERA5':
          PTOP_LOW = 0.8*psfc # these are arrays
          PTOP_MID = 0.45*psfc
          PTOP_HIGH = PTOP_HIGH_UPP * np.ones_like(psfc)
       elif layerDefinitions.upper().strip() == 'UPP':
          PTOP_LOW = PTOP_LOW_UPP # these are constants
          PTOP_MID = PTOP_MID_UPP
          PTOP_HIGH = PTOP_HIGH_UPP
       else:
          print('layerDefinitions = ',layerDefinitions,'is invalid. exit')
          sys.exit()
```

(continues on next page)

(continued from previous page)

```
        fcstLow,fcstMid,fcstHigh,fcstTotCldFra = getFcstCloudFrac(fcstCldfra,pressure,psfc,
↪layerDefinitions) # get low/mid/high/total forecast cloud fractions for each ob
        nc_fid2.close()

        # Modify QC data based on correspondence between forecast and obs. qcData used to␣
↪select good data later
        # It's possible that there are multiple forecast layers, such that fcstLow,fcstMid,
↪fcstHigh are all > $cldfraThresh
        # However, GOES-16 CTP doesn't really account for layering.  So, we need to remove␣
↪layered clouds from the forecast,
        #   focusing only on the layers that we asked for when doing {low,mid,high}Only␣
↪conditions
        # The "|" is symbol for "np.logcal_or"
        yes = 2.0
        no  = 0.0
        cldfraThresh = 20.0 # percent
        if qcData.shape == fcstTotCldFra.shape == thisGOESData.shape:  # these should all␣
↪match
           print('Using condition ',condition,'for ABI/AHI')

           # Note that "&" is "np.logical_and" for boolean (true/false) quantities.
           # Thus, each condition should be enclosed in parentheses
           if   condition.lower().strip() == 'clearOnly'.lower():  # clear in both forecast␣
↪and obs
              qcData = np.where( (fcstTotCldFra < cldfraThresh)  & (thisGOESData <= 0.0),␣
↪qcData, missing_values)
           elif condition.lower().strip() == 'cloudyOnly'.lower(): # cloudy in both␣
↪forecast and obs
              qcData = np.where( (fcstTotCldFra >= cldfraThresh) & (thisGOESData > 0.0),␣
↪qcData, missing_values)
           elif condition.lower().strip() == 'lowOnly'.lower(): # low clouds in both␣
↪forecast and obs
              fcstLow = np.where( (fcstMid >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstLow) # remove mid, high
              qcData = np.where( (fcstLow >= cldfraThresh) & ( thisGOESData >= PTOP_LOW),␣
↪qcData, missing_values)
           elif condition.lower().strip() == 'midOnly'.lower(): # mid clouds in both␣
↪forecast and obs
              fcstMid = np.where( (fcstLow >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstMid) # remove low, high
              qcData = np.where( (fcstMid >= cldfraThresh) & (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID),   qcData, missing_values)
           elif condition.lower().strip() == 'highOnly'.lower(): # high clouds in both␣
↪forecast and obs
              fcstHigh = np.where( (fcstLow >= cldfraThresh) | ( fcstMid >= cldfraThresh),␣
```

(continues on next page)

(continued from previous page)

```
↪missing_values, fcstHigh) # remove mid, high
              qcData = np.where( (fcstHigh >= cldfraThresh) & (thisGOESData <  PTOP_MID) &␣
↪(thisGOESData >= PTOP_HIGH), qcData, missing_values)
           elif condition.lower().strip() == 'fcstLow'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
              qcData = np.where( fcstLow >= cldfraThresh , qcData, missing_values)
           elif condition.lower().strip() == 'fcstMid'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
              qcData = np.where( fcstMid >= cldfraThresh , qcData, missing_values)
           elif condition.lower().strip() == 'fcstHigh'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
              qcData = np.where( fcstHigh >= cldfraThresh , qcData, missing_values)
           elif condition.lower().strip() == 'cloudEventLow'.lower():
              if dataSource == 1: this_var = np.where( fcstLow      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
              if dataSource == 2: this_var = np.where( thisGOESData >= PTOP_LOW, yes, no )
           elif condition.lower().strip() == 'cloudEventMid'.lower():
              if dataSource == 1: this_var = np.where( fcstMid      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
              if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID), yes, no )
           elif condition.lower().strip() == 'cloudEventHigh'.lower():
              if dataSource == 1: this_var = np.where( fcstHigh     >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
              if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_MID) &␣
↪(thisGOESData >= PTOP_HIGH), yes, no )
           elif condition.lower().strip() == 'cloudEventTot'.lower():
              if dataSource == 1: this_var = np.where( fcstTotCldFra >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
              if dataSource == 2: this_var = np.where( thisGOESData  > 0.0, yes, no )
           elif condition.lower().strip() == 'all':
              print("not doing any conditional verification or stratifying by event")
           else:
              print("condition = ",condition," not recognized.")
              sys.exit()
           #elif condition.lower().strip() == '4x4table'.lower():
             #if dataSource == 1:
             #   this_var = np.where( fcstLow >= cldfraThresh, yesLow, no )
             #   this_var = this_var + np.where( fcstMid >= cldfraThresh, yesMid, no )
             #   this_var = this_var + np.where( fcstHigh >= cldfraThresh, yesHigh, no )
           print('number removed = ', (qcData==missing_values).sum())
          #print('number passed   = ', qcData.shape[0] - (qcData==missing_values).sum())
        else:
           print('shape mismatch')
           return -99999, -99999
```

(continues on next page)

(continued from previous page)

```
      # Append to arrays
      allData.append(this_var)
      allDataQC.append(qcData)

      nc_fid.close() # done with the file, so close it before going to next file in loop

   # We're now all done looping over the individul files

   # Get the indices with acceptable QC
   allQC = np.concatenate(allDataQC) # Put list of numpy arrays into a single long 1-D numpy␣
↪array.  All QC data.
   idx = np.where(allQC==0) # returns indices

   # Now get all the forecast/observed brightness temperature data with acceptable QC
   this_var = np.concatenate(allData)[idx] # Put list of numpy arrays into a single long 1-D␣
↪numpy array. This is all the forecast/obs data with good QC
   numObs = this_var.shape[0] # number of points with good QC for this channel
   print('Number of obs :',numObs)

   # Assume all the points actually fit into a square grid. Get the side of the square (use␣
↪ceil to round up)
   if numObs > 0:
      l = np.ceil(np.sqrt(numObs)).astype('int') # Length of the side of the square

      # Make an array that can be reshaped into the square
      raw_data1D = np.full(l*l,np.nan) # Initialize 1D array of length l**2 to np.nan
      raw_data1D[0:numObs] = this_var[:] # Fill data to the extent possible. There will be␣
↪some np.nan values at the end
      raw_data = np.reshape(raw_data1D,(l,l)) # Reshape into "square grid"

      raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

      met_data=raw_data.astype(float) # Give MET this info

      # Now need to tell MET the "grid" for the data
      # Make a fake lat/lon grid going from 0.0 to 50.0 degrees, with the interval␣
↪determined by number of points
      griddedDatasets[source]['latDef'][0] = 0.0 # starting point
      griddedDatasets[source]['latDef'][1] = np.diff(np.linspace(0,50,l)).round(6)[0] #␣
↪interval (degrees)
      griddedDatasets[source]['latDef'][2] = int(l) # number of points
      griddedDatasets[source]['lonDef'][0:3] = griddedDatasets[source]['latDef']
```

(continues on next page)

(continued from previous page)

```
      gridInfo = getGridInfo(source, griddedDatasets[source]['gridType']) # 'LatLon' gridType
      return met_data, gridInfo

   else:
      return -99999, -99999

###########
def getGridInfo(source,gridType):

   if gridType == 'LatLon':
      latDef = griddedDatasets[source]['latDef']
      lonDef = griddedDatasets[source]['lonDef']
      gridInfo = {
         'type':      gridType,
         'name':      source,
         'lat_ll':    latDef[0], #-90.000,
         'lon_ll':    lonDef[0], #-180.000,
         'delta_lat': latDef[1], #0.5000,
         'delta_lon': lonDef[1], #0.625,
         'Nlat':      latDef[2], #361,
         'Nlon':      lonDef[2], #576,
      }
   elif gridType == 'Gaussian':
      gridInfo = {
        'type':     gridType,
        'name':     source,
        'nx':       griddedDatasets[source]['nx'],
        'ny':       griddedDatasets[source]['ny'],
        'lon_zero': griddedDatasets[source]['lon_zero'],
      }

   return gridInfo

def getAttrArray(source,variable,initTime,validTime):

   init = dt.datetime.strptime(initTime,"%Y%m%d%H")
   valid = dt.datetime.strptime(validTime,"%Y%m%d%H")
   lead, rem = divmod((valid-init).total_seconds(), 3600)

   attrs = {

      'valid': valid.strftime("%Y%m%d_%H%M%S"),
      'init':  init.strftime("%Y%m%d_%H%M%S"),
      'lead':  str(int(lead)),
      'accum': '000000',
```

(continues on next page)

(continued from previous page)

```
      'name':      variable,  #'MERRA2_Cloud_Percentage'
      'long_name': variable,  #'Cloud Percentage Levels',
      'level':     'ALL',
      'units':     verifVariables[variable]['units'],

      'grid': getGridInfo(source,griddedDatasets[source]['gridType'])
   }

   #print(attrs)
   #print(griddedDatasets[source])

   return attrs

######## END FUNCTIONS   ##########


#if __name__ == "__main__":
dataFile, dataSource, variable, i_date, v_date, flag = sys.argv[1].split(":")
met_data = getDataArray(dataFile,dataSource,variable,flag)
attrs = getAttrArray(dataSource,variable,i_date,v_date)
print(attrs)
```

Running METplus

Pass the use case configuration file to the run_metplus.py script along with any user-specific system configuration files if desired:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/clouds/GridStat_fcstMPAS_
↪obsSATCORPS_lowAndTotalCloudFrac.conf /path/to/user_system.conf
```

See *Running METplus* (page 26) for more information.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/clouds/GridStat_fcstMPAS_obsSATCORPS_lowAndTotalCloudFrac (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_MPAS_F36_CloudFracs_360000L_20200724_120000V_pairs.nc
- grid_stat_MPAS_F36_CloudFracs_360000L_20200724_120000V.stat

- grid_stat_MPAS_F36_CloudFracs_NBR_360000L_20200724_120000V_pairs.nc
- grid_stat_MPAS_F36_CloudFracs_NBR_360000L_20200724_120000V.stat
- grid_stat_MPAS_F36_CloudFracs_PROB_360000L_20200724_120000V_pairs.nc
- grid_stat_MPAS_F36_CloudFracs_PROB_360000L_20200724_120000V.stat

Keywords

Note:

- GridStatToolUseCase
- NetCDFFileUseCase
- CloudsAppUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/clouds-GridStat_fcstMPAS_obsSATCORPS_lowAndTotalCloudFrac.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.3.6 GridStat: Cloud Fractions with Neighborhood and Probabilities

model_applications/clouds/GridStat_fcstMPAS_obsMERRA2_lowAndTotalCloudFrac.conf

Scientific Objective

This use case captures various statistical measures of two model comparisons for low and total cloud fraction with different neighborhood and probability settings for internal model metrics and to aid in future model updates

Datasets

Forecast: Model for Prediction Across Scales (MPAS)
Observations: Modern-Era Retrospective analysis for Research and Applications, Version 2 (MERRA2)
Grid: GPP 17km masking region

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

METplus Components

This use case utilizes Python Embedding, which is called using the PYTHON_NUMPY keyword in the forecast and observation input template settings. The same Python script processes both forecast and observation datasets. Two separate forecast fields are verified against two respective observation fields, with the Python script being passed the input file, the model name, the variable name being analyzed, the initialization and valid times, and a flag to indicate if the field passed is observation or forecast. This process is repeated with 3 instance names to GridStat, each with a different setting for regridding, neighborhood evaluation, thresholding, output line types, and output prefix names.

METplus Workflow

GridStat is the only MET tool called in this example. It processes the following run time:

Init: 2020-07-23 00Z
Forecast lead: 36 hour

Because instance names are used, GridStat will run 3 times for this 1 initalization time.

METplus Configuration

METplus first loads the default configuration file found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line: parm/use_cases/model_applications/clouds/GridStat_fcstMPAS_obsMERRA2_lowAndTotalCloudFrac.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/clouds/GridStat_
↪fcstMPAS_obsMERRA2_lowAndTotalCloudFrac.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

# ###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###
```

(continues on next page)

(continued from previous page)

```
PROCESS_LIST = GridStat, GridStat(nbr), GridStat(prob)

###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2020072300
INIT_END=2020072300
INIT_INCREMENT = 12H

LEAD_SEQ = 36

LOOP_ORDER = times

###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR =
FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_GRID_STAT_INPUT_DIR =
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_CLIMO_MEAN_INPUT_DIR =
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/clouds/GridStat_fcstMPAS_obsMERRA2_
↪lowAndTotalCloudFrac
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_OUTPUT_TEMPLATE =


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = MPAS
OBTYPE = MERRA2

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/clouds/GridStat_fcstMPAS_obsMERRA2_
↪lowAndTotalCloudFrac

FCST_VAR1_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstMPAS_obsMERRA2_lowAndTotalCloudFrac/diag.{valid?fmt=%Y-%m-%d_%H}.00.00_latlon.
↪nc:{MODEL}:totalCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:1
FCST_VAR1_LEVELS =
FCST_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0

OBS_VAR1_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstMPAS_obsMERRA2_lowAndTotalCloudFrac/MERRA2_400.tavg1_2d_rad_Nx.{valid?fmt=%Y%m
↪%d}.nc4:{OBTYPE}:totalCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:2
OBS_VAR1_LEVELS =
OBS_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0


FCST_VAR2_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstMPAS_obsMERRA2_lowAndTotalCloudFrac/diag.{valid?fmt=%Y-%m-%d_%H}.00.00_latlon.
↪nc:{MODEL}:lowCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:1
FCST_VAR2_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0

OBS_VAR2_NAME = {CONFIG_DIR}/read_input_data.py {INPUT_BASE}/model_applications/clouds/
↪GridStat_fcstMPAS_obsMERRA2_lowAndTotalCloudFrac/MERRA2_400.tavg1_2d_rad_Nx.{valid?fmt=%Y%m
↪%d}.nc4:{OBTYPE}:lowCloudFrac:{init?fmt=%Y%m%d%H}:{valid?fmt=%Y%m%d%H}:2
OBS_VAR2_THRESH =  gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
```

(continues on next page)

(continued from previous page)

```
###

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

GRID_STAT_REGRID_TO_GRID = FCST
GRID_STAT_REGRID_METHOD = BILIN
GRID_STAT_REGRID_WIDTH = 2


GRID_STAT_DESC =

FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0
OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_ONCE_PER_FIELD = False

FCST_IS_PROB = false

FCST_GRID_STAT_PROB_THRESH = ==0.1

OBS_IS_PROB = false

OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudFracs

GRID_STAT_OUTPUT_FLAG_FHO = STAT
GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_CNT = STAT
GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE


GRID_STAT_MASK_POLY = {INPUT_BASE}/model_applications/clouds/GridStat_fcstMPAS_obsMERRA2_
↪lowAndTotalCloudFrac/GPP_17km_60S_60N_mask.nc


[nbr]

FCST_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SFP20, >SFP30, >SFP40, >SFP50, >SFP60, >SFP70, >SFP80

OBS_VAR1_THRESH = gt0, lt10.0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.
↪0, ge90.0, >SOP20, >SOP30, >SOP40, >SOP50, >SOP60, >SOP70, >SOP80

GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9
GRID_STAT_NEIGHBORHOOD_SHAPE = CIRCLE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >0.0

GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = NONE
GRID_STAT_OUTPUT_FLAG_CTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = NONE
GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTC = STAT
GRID_STAT_OUTPUT_FLAG_NBRCTS = STAT
GRID_STAT_OUTPUT_FLAG_NBRCNT = STAT
GRID_STAT_OUTPUT_FLAG_GRAD = NONE

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_NBRHD = TRUE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudFracs_NBR


[prob]

FCST_IS_PROB = TRUE

FCST_VAR1_THRESH = >0.1, >0.2, >0.3, >0.4, >0.5, >0.6, >0.7, >0.8, >0.9, >1.0
```

(continues on next page)

(continued from previous page)

```
OBS_VAR1_THRESH = gt0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.0, ge90.0

FCST_VAR2_THRESH = >0.1, >0.2, >0.3, >0.4, >0.5, >0.6, >0.7, >0.8, >0.9, >1.0
OBS_VAR2_THRESH = gt0, ge10.0, ge20.0, ge30.0, ge40.0, ge50.0, ge60.0, ge70.0, ge80.0, ge90.0

GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9
GRID_STAT_NEIGHBORHOOD_SHAPE = CIRCLE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >0.0

GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = NONE
GRID_STAT_OUTPUT_FLAG_CTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = NONE
GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTC = NONE
GRID_STAT_OUTPUT_FLAG_NBRCTS = NONE
GRID_STAT_OUTPUT_FLAG_NBRCNT = NONE
GRID_STAT_OUTPUT_FLAG_GRAD = NONE
GRID_STAT_OUTPUT_FLAG_PCT = STAT
GRID_STAT_OUTPUT_FLAG_PSTD = STAT
GRID_STAT_OUTPUT_FLAG_PJC = STAT
GRID_STAT_OUTPUT_FLAG_PRC = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE
GRID_STAT_NC_PAIRS_FLAG_NBRHD = FALSE
GRID_STAT_NC_PAIRS_FLAG_GRADIENT = TRUE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE

GRID_STAT_OUTPUT_PREFIX = {MODEL}_to_{OBTYPE}_F{lead?fmt=%H}_CloudFracs_PROB
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on

the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
```

(continues on next page)

(continued from previous page)

```
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
```

(continues on next page)

(continued from previous page)

```
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)
```

(continues on next page)

(continued from previous page)

```
//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case utilizes 1 Python script to read and process both forecast and observation fields. parm/use_cases/model_applications/clouds/GridStat_fcstMPAS_obsMERRA2_lowAndTotalCloudFrac/read_input_dat

```
#this code was provided by Craig Schwartz
#and is largely unaltered from its original
#function.

#from __future__ import print_function
import os
import sys
import numpy as np
import datetime as dt
from netCDF4 import Dataset  # http://code.google.com/p/netcdf4-python/
from scipy.interpolate import NearestNDInterpolator, LinearNDInterpolator
#### for Plotting
import matplotlib.cm as cm
import matplotlib.axes as maxes
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
#from mpl_toolkits.basemap import Basemap
import fnmatch
import pygrib
import pickle as pk
#####
```

(continues on next page)

(continued from previous page)

```
########################################

missing_values = -9999.0  # for MET

# UPP top layer bounds (Pa) for cloud layers
PTOP_LOW_UPP  = 64200. # low for > 64200 Pa
PTOP_MID_UPP  = 35000. # mid between 35000-64200 Pa
PTOP_HIGH_UPP = 15000. # high between 15000-35000 Pa

# Values for 4 x 4 contingency table
Na, Nb, Nc, Nd = 1, 2, 3, 4
Ne, Nf, Ng, Nh = 5, 6, 7, 8
Ni, Nj, Nk, Nl = 9, 10, 11, 12
Nm, Nn, No, Np = 13, 14, 15, 16

# Notes:
# 1) Entry for 'point' is for point-to-point comparison and is all dummy data (except for␣
↪gridType) that is overwritten by point2point
# 2) ERA5 on NCAR CISL RDA changed at some point.  Old is ERA5_2017 (not used anymore), new␣
↪is ERA5, which we'll use for 2020 data
griddedDatasets =  {
   'MERRA2'    : { 'gridType':'LatLon', 'latVar':'lat',      'latDef':[-90.0,0.50,361], 'lonVar
↪':'lon',        'lonDef':[-180.0,0.625,576],   'flipY':True, 'ftype':'nc'},
   'SATCORPS' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[-180.0,0.3125,1152], 'flipY':False, 'ftype':'nc' },
   'ERA5_2017': { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.7848769072,0.
↪281016829130516,640], 'lonVar':'longitude', 'lonDef':[0.0,0.28125,1280], 'flipY':False,
↪'ftype':'nc' },
   'ERA5'      : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'GFS'       : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'GALWEM'    : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':True, 'ftype':'grib'},
   'GALWEM17' : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-89.921875,0.156250,
↪1152], 'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536], 'flipY':False, 'ftype':
↪'grib'},
   'WWMCA'     : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721], 'lonVar
↪':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'grib'},
   'MPAS'      : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.25,721],
↪'lonVar':'longitude',  'lonDef':[0.0,0.25,1440],   'flipY':False, 'ftype':'nc'},
   'SAT_WWMCA_MEAN' : { 'gridType':'LatLon', 'latVar':'lat','latDef':[-90.0,0.25,721],
↪'lonVar':'lon', 'lonDef':[0.0,0.25,1440], 'flipY':False, 'ftype':'nc' },
   'point'     : { 'gridType':'LatLon', 'latVar':'latitude','latDef':[-90.0,0.156250,1152],
↪'lonVar':'longitude',  'lonDef':[0.117187,0.234375,1536],   'flipY':False, 'ftype':'nc'},
```

(continues on next page)

(continued from previous page)

```
}
   #TODO:Correct one, but MET can ingest a Gaussian grid only in Grib2 format (from Randy B.)
   #'ERA5'     : { 'gridType':'Gaussian', 'nx':1280, 'ny':640, 'lon_zero':0, 'latVar':
↪'latitude', 'lonVar':'longitude', 'flipY':False, },

#GALWEM, both 17-km and 0.25-degree
lowCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':3,
↪'typeOfFirstFixedSurface':10, 'shortName':'lcc' }
midCloudFrac_GALWEM  =  { 'parameterCategory':6, 'parameterNumber':4,
↪'typeOfFirstFixedSurface':10, 'shortName':'mcc' }
highCloudFrac_GALWEM =  { 'parameterCategory':6, 'parameterNumber':5,
↪'typeOfFirstFixedSurface':10, 'shortName':'hcc' }
totalCloudFrac_GALWEM = { 'parameterCategory':6, 'parameterNumber':1,
↪'typeOfFirstFixedSurface':10, 'shortName':'tcc' }
cloudTopHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':12,
↪'typeOfFirstFixedSurface':3, 'shortName':'cdct' }
cloudBaseHeight_GALWEM =  { 'parameterCategory':6, 'parameterNumber':11,
↪'typeOfFirstFixedSurface':2, 'shortName':'cdcb' }

#GFS
lowCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':214, 'shortName':'tcc' }
midCloudFrac_GFS  =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':224, 'shortName':'tcc' }
highCloudFrac_GFS =  { 'parameterCategory':6, 'parameterNumber':1, 'typeOfFirstFixedSurface
↪':234, 'shortName':'tcc' }

#WWMCA
totalCloudFrac_WWMCA  = { 'parameterName':71, 'typeOfLevel':'entireAtmosphere', 'level':0 }

cloudTopHeightLev1_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':1 }
cloudTopHeightLev2_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':2 }
cloudTopHeightLev3_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':3 }
cloudTopHeightLev4_WWMCA  = { 'parameterName':228, 'typeOfLevel':'hybrid', 'level':4 }
cloudTopHeight_WWMCA      = [ cloudTopHeightLev1_WWMCA, cloudTopHeightLev2_WWMCA,␣
↪cloudTopHeightLev3_WWMCA, cloudTopHeightLev4_WWMCA ]

cloudBaseHeightLev1_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':1 }
cloudBaseHeightLev2_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':2 }
cloudBaseHeightLev3_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':3 }
cloudBaseHeightLev4_WWMCA  = { 'parameterName':227, 'typeOfLevel':'hybrid', 'level':4 }
cloudBaseHeight_WWMCA      = [ cloudBaseHeightLev1_WWMCA, cloudBaseHeightLev2_WWMCA,␣
↪cloudBaseHeightLev3_WWMCA, cloudBaseHeightLev4_WWMCA ]

verifVariablesModel = {
```

(continues on next page)

(continued from previous page)

```
    'binaryCloud'    :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
    'totalCloudFrac' :  {'GFS':[''], 'GALWEM17':[totalCloudFrac_GALWEM],  'GALWEM
↪':[totalCloudFrac_GALWEM], 'MPAS':['cldfrac_tot_UM_rand']},
    'lowCloudFrac'   :  {'GFS':[lowCloudFrac_GFS], 'GALWEM17':[lowCloudFrac_GALWEM], 'GALWEM
↪':[lowCloudFrac_GALWEM], 'MPAS':['cldfrac_low_UM']},
    'midCloudFrac'   :  {'GFS':[midCloudFrac_GFS], 'GALWEM17':[midCloudFrac_GALWEM], 'GALWEM
↪':[midCloudFrac_GALWEM], 'MPAS':['cldfrac_mid_UM']},
    'highCloudFrac'  :  {'GFS':[highCloudFrac_GFS], 'GALWEM17':[highCloudFrac_GALWEM],
↪'GALWEM':[highCloudFrac_GALWEM], 'MPAS':['cldfrac_high_UM']},
    'cloudTopHeight' :  {'GFS':['']               , 'GALWEM17':[cloudTopHeight_GALWEM],
↪'GALWEM':[cloudTopHeight_GALWEM], 'MPAS':['cldht_top_UM']},
    'cloudBaseHeight' : {'GFS':['']               , 'GALWEM17':[cloudBaseHeight_GALWEM],
↪'GALWEM':[cloudBaseHeight_GALWEM], 'MPAS':['cldht_base_UM']},
}

cloudFracCatThresholds = '>0, <10.0, >=10.0, >=20.0, >=30.0, >=40.0, >=50.0, >=60.0, >=70.0,␣
↪>=80.0, >=90.0' # MET format string
brightnessTempThresholds = '<280.0, <275.0, <273.15, <270.0, <265.0, <260.0, <255.0, <250.0,
↪<245.0, <240.0, <235.0, <230.0, <225.0, <220.0, <215.0, <210.0, <=SFP1, <=SFP5, <=SFP10,
↪<=SFP25, <=SFP50, >=SFP50, >=SFP75, >=SFP90, >=SFP95, >=SFP99'
verifVariables = {
   'binaryCloud'    : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['TCC'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'NA',  'thresholds':'>0.0', 'interpMethod':'nearest' },
   'totalCloudFrac' : { 'MERRA2':['CLDTOT'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['tcc'], 'WWMCA':[totalCloudFrac_WWMCA], 'SAT_WWMCA_MEAN':['Mean_WWMCA_SATCORPS'],
↪'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin' },
   'lowCloudFrac'   : { 'MERRA2':['CLDLOW'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['lcc'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'midCloudFrac'   : { 'MERRA2':['CLDMID'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['MCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'highCloudFrac'  : { 'MERRA2':['CLDHGH'], 'SATCORPS':['cloud_percentage_level'],
↪'ERA5':['HCC'], 'units':'%',   'thresholds':cloudFracCatThresholds, 'interpMethod':'bilin'␣
↪},
   'cloudTopTemp'   : { 'MERRA2':['CLDTMP'], 'SATCORPS':['cloud_temperature_top_level'],
↪'ERA5':['']   , 'units':'K',   'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopPres'   : { 'MERRA2':['CLDPRS'], 'SATCORPS':['cloud_pressure_top_level'],
↪'ERA5':['']   , 'units':'hPa', 'thresholds':'NA', 'interpMethod':'bilin'},
   'cloudTopHeight' : { 'MERRA2':['']      , 'SATCORPS':['cloud_height_top_level'],
↪'ERA5':['']   , 'WWMCA':cloudTopHeight_WWMCA,  'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
   'cloudBaseHeight': { 'MERRA2':['']      , 'SATCORPS':['cloud_height_base_level'],
```

(continues on next page)

(continued from previous page)

```
↪'ERA5':['cbh'], 'WWMCA':cloudBaseHeight_WWMCA, 'units':'m',   'thresholds':'NA',
↪'interpMethod':'nearest'},
   'cloudCeiling'   : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'m',   'thresholds':'NA', 'interpMethod':'bilin'},
   'brightnessTemp' : { 'MERRA2':['']      , 'SATCORPS':[''],
↪'ERA5':['']   , 'units':'K',   'thresholds':brightnessTempThresholds, 'interpMethod':'bilin
↪'},
}

# Combine the two dictionaries
# Only reason verifVariablesModel exists is just for space--verifVaribles gets too long if␣
↪we keep adding more datasets
for key in verifVariablesModel.keys():
  x = verifVariablesModel[key]
  for key1 in x.keys():
     verifVariables[key][key1] = x[key1]

#f = '/glade/u/home/schwartz/cloud_verification/GFS_grib_0.25deg/2018112412/gfs.0p25.
↪2018112412.f006.grib2'
#grbs = pygrib.open(f)
#idx = pygrib.index(f,'parameterCategory','parameterNumber','typeOfFirstFixedSurface')
#model = 'GFS'
#variable = 'totCloudCover'
#x = verifVariablesModel[variable][model] # returns a list, whose ith element is a dictionary
# e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
#idx(parameterCategory=x[0]['parameterCategory'],parameterNumber=x[0]['parameterNumber'],
↪typeOfFirstFixedSurface=x[0]['typeOfFirstFixedSurface'])

# to read in an environmental variable
#x = os.getenv('a') # probably type string no matter what

###########

def getThreshold(variable):
   x = verifVariables[variable]['thresholds']
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getInterpMethod(variable):
   x = verifVariables[variable]['interpMethod'].upper()
   print(x) # needed for python 3 to read variable into csh variable
   return x

def getTotalCloudFrac(source,data):
   if source == 'SATCORPS':
```

(continues on next page)

(continued from previous page)

```
    # x = data[0][0,:,:,0] * 1.0E-2  # scaling
      x = (data[0][0,:,:,1]  + data[0][0,:,:,2] + data[0][0,:,:,3])*1.0E-2  # scaling
   #  y = data[0]
   #  x = np.sum( y[:,:,:,1:4],axis=3)
   elif source == 'MERRA2':
#      x = ( data[0][0,:,:]+data[1][0,:,:]+data[2][0,:,:] ) *100.0 # the ith element of data␣
↪is a numpy array
      x = data[0][0,:,:] * 100.0 # the ith element of data is a numpy array
      print(x.min(), x.max())
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
      x = data[0][0,:,:] * 100.0
   elif source == 'SAT_WWMCA_MEAN':
      x = data[0][0,:,:] # already in %
   else:
      x = data[0]

   # This next line is WRONG.
   # Missing should be set to missing
   # Then, the non-missing values are 1s and 0s
   #output = np.where(x > 0.0, x, 0.0)
   #output = np.where(x < 0.0, -9999.0, x) # missing. currently used for SATCORPS

   x = np.where( x < 0.0  , 0.0,   x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%
   return x

def getBinaryCloud(source,data):
   y = getTotalCloudFrac(source,data)
   # keep NaNs as is, but then set everything else to either 100% or 0%
   x = np.where( np.isnan(y), y, np.where(y > 0.0, 100.0, 0.0) )
   return x

def getLayerCloudFrac(source,data,layer):
   if source == 'SATCORPS':
      if layer.lower().strip() == 'low'  : i = 1
      if layer.lower().strip() == 'mid'  : i = 2
      if layer.lower().strip() == 'high' : i = 3
      x = data[0][0,:,:,i] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 100.0
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] * 100.0
```

(continues on next page)

(continued from previous page)

```
      except: x = data[0][0,:,:] * 100.0
   elif source == 'MPAS':
      x = data[0][0,:,:] * 100.0
   else:
      x = data[0]

   x = np.where( x < 0.0, 0.0, x) # Force negative values to zero
   x = np.where( x > 100.0, 100.0, x) # Force values > 100% to 100%

   return x

def getCloudTopTemp(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E-2  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
      x = data[0]
   return x

def getCloudTopPres(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E-1  # scaling
   elif source == 'MERRA2':
      x = data[0][0,:,:] * 1.0E-2  # scaling [Pa] -> [hPa]
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   else:
      x = data[0]
   return x

def getCloudTopHeight(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
```

(continues on next page)

(continued from previous page)

```
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
         print('error with WWMCA Cloud top height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmax(tmp,axis=0) # get maximum cloud top height across all layers
   else:
      x = data[0]

   # Eliminate unphysical values (assume cloud top shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y

def getCloudBaseHeight(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0] * 1.0E+1  # scaling to [meters]
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:]
      except: x = data[0][0,:,:]
   elif source == 'GALWEM17':
      x = data[0] * 1000.0 * 0.3048  # kilofeet -> meters
   elif source == 'MPAS':
      x = data[0][0,:,:] # already in meters
   elif source == 'WWMCA':
      # data is a list (should be length 4)
      if len(data) != 4:
         print('error with WWMCA Cloud base height')
         sys.exit()
      tmp = np.array(data) # already in meters
      tmp = np.where( tmp <= 0, np.nan, tmp) # replace 0 or negative values with NAN
      x = np.nanmin(tmp,axis=0) # get lowest cloud base over all layers
   else:
      x = data[0]

   # Eliminate unphysical values (assume cloud base shouldn't be > 50000 meters)
   y = np.where( x > 50000.0 , np.nan, x )

   return y
```

(continues on next page)

(continued from previous page)

```
def getCloudCeiling(source,data):
   if source == 'SATCORPS':
      x = data[0][0,:,:,0]   #TBD
   elif source == 'MERRA2':
      x = data[0][0,:,:]     #TBD
   elif source == 'ERA5':
      try:    x = data[0][0,0,:,:] # TBD
      except: x = data[0][0,:,:]
   return x

# add other functions for different variables

###########

def getDataArray(inputFile,source,variable,dataSource):
   # 1) inputFile:  File name--either observations or forecast
   # 2) source:     Obsevation source (e.g., MERRA, SATCORP, etc.)
   # 3) variable:   Variable to verify
   # 4) dataSource: If 1, process forecast file. If 2 process obs file.

#   # specifying names here temporarily. file names should be passed in to python from shell␣
↪script
#   if source == 'merra':      nc_file = '/gpfs/fs1/scratch/schwartz/MERRA/MERRA2_400.tavg1_
↪2d_rad_Nx.20181101.nc4'
#   elif source == 'satcorp':  nc_file = '/glade/scratch/bjung/met/test_satcorps/GEO-MRGD.
↪2018334.0000.GRID.NC'
#   elif source == 'era5':     nc_file = '/glade/scratch/bjung/met/test_era5/e5.oper.fc.sfc.
↪instan.128_164_tcc.regn320sc.2018111606_2018120112.nc'

   source = source.upper().strip()  # Force uppercase and get rid of blank spaces, for safety

   print('dataSource = ',dataSource)

   ftype = griddedDatasets[source]['ftype'].lower().strip()

   # Get file handle
   if ftype == 'nc':
      nc_fid = Dataset(inputFile, "r", format="NETCDF4")
      #nc_fid.set_auto_scale(True)
   elif ftype == 'grib':
      if source == 'WWMCA':
        idx = pygrib.index(inputFile,'parameterName','typeOfLevel','level')
      else:
        idx = pygrib.index(inputFile,'parameterCategory','parameterNumber',
```

(continues on next page)

(continued from previous page)

```
↪'typeOfFirstFixedSurface')

   # dataSource == 1 means forecast, 2 means obs
#  if dataSource == 1: varsToRead = verifVariablesModel[variable][source] # if ftype == 'grib
↪', returns a list whose ith element is a dictionary. otherwise, just a list
#  if dataSource == 2: varsToRead = verifVariables[variable][source] # returns a list
   varsToRead = verifVariables[variable][source] # if ftype == 'grib', returns a list whose␣
↪ith element is a dictionary. otherwise, just a list

   print('Trying to read ',inputFile)

   # Get lat/lon information--currently not used
  #latVar = griddedDatasets[source]['latVar']
  #lonVar = griddedDatasets[source]['lonVar']
  #lats = np.array(nc_fid.variables[latVar][:])   # extract/copy the data
  #lons = np.array(nc_fid.variables[lonVar][:] )

   #print(lats.max())
   #print(lons.max())

   # one way to deal with scale factors
   # probably using something like nc_fid.set_auto_scale(True) is better...
  #latMax = lats.max()
  #while latMax > 90.0:
  #   lons = lons * 0.1
  #   lats = lats * 0.1
  #   latMax = lats.max()

   # get data
   data = []
   for v in varsToRead:
      if ftype == 'grib':
         if source == 'WWMCA':
           x = idx(parameterName=v['parameterName'],typeOfLevel=v['typeOfLevel'],level=v[
↪'level'])[0] # by getting element 0, you get a pygrib message
         else:
            # e.g., idx(parameterCategory=6,parameterNumber=1,typeOfFirstFixedSurface=234)
            if ( variable == 'cloudTopHeight' or variable == 'cloudBaseHeight') and source␣
↪== 'GALWEM17':
               x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[1] # by getting␣
↪element 1, you get a pygrib message
            else:
               x = idx(parameterCategory=v['parameterCategory'],parameterNumber=v[
↪'parameterNumber'],typeOfFirstFixedSurface=v['typeOfFirstFixedSurface'])[0] # by getting␣
```

(continues on next page)

(continued from previous page)

```
↪element 0, you get a pygrib message
            if x.shortName != v['shortName']: print('Name mismatch!')
            #ADDED BY JOHN O
            print(x)
            print('Reading ', x.shortName, 'at level ', x.typeOfFirstFixedSurface)
         read_var = x.values # same x.data()[0]
         read_missing = x.missingValue
         print('missing value = ',read_missing)

         # The missing value (read_missing) for GALWEM17 and GALWEM cloud base/height is␣
↪9999, which is not the best choice because
         # those could be actual values. So we need to use the masked array part (below) to␣
↪handle which
         # values are missing.  We also set read_missing to something unphysical to␣
↪essentially disable it.
         # Finally, if we don't change the 'missingValue' property in the GRIB2 file we are␣
↪eventually outputting,
         # the bitmap will get all messed up, because it will be based on 9999 instead of
↪$missing_values
         if variable == 'cloudTopHeight' or variable == 'cloudBaseHeight':
            read_missing = -9999.
            x['missingValue'] = read_missing
            if source == 'GALWEM17':
               #These are masked numpy arrays, with mask = True where there is a missing␣
↪value (no cloud)
               #Use np.ma.filled to create an ndarray where mask = True values are set to np.
↪nan
               read_var = np.ma.filled(read_var.astype(read_var.dtype), np.nan)
      elif ftype == 'nc':
         read_var = nc_fid.variables[v]         # extract/copy the data
         try:
            read_missing = read_var.missing_value  # get variable attributes. Each dataset␣
↪has own missing values.
         except:
            read_missing = -9999. # set a default missing value. probably only need to do␣
↪this for MPAS

      print('Reading ', v)

      this_var = np.array( read_var )        # to numpy array
     #print(read_missing, np.nan)
      this_var = np.where( this_var==read_missing, np.nan, this_var )
     #print(this_var.shape)
      data.append(this_var) # ith element of the list is a NUMPY ARRAY for the ith variable
     #print(type(this_var))
```

(continues on next page)

(continued from previous page)

```
    #print(type(data))

  # Call a function to get the variable of interest.
  # Add a new function for each variable
  if variable == 'binaryCloud':     raw_data = getBinaryCloud(source,data)
  if variable == 'totalCloudFrac':  raw_data = getTotalCloudFrac(source,data)
  if variable == 'lowCloudFrac':    raw_data = getLayerCloudFrac(source,data,'low')
  if variable == 'midCloudFrac':    raw_data = getLayerCloudFrac(source,data,'mid')
  if variable == 'highCloudFrac':   raw_data = getLayerCloudFrac(source,data,'high')
  if variable == 'cloudTopTemp':    raw_data = getCloudTopTemp(source,data)
  if variable == 'cloudTopPres':    raw_data = getCloudTopPres(source,data)
  if variable == 'cloudTopHeight':  raw_data = getCloudTopHeight(source,data)
  if variable == 'cloudBaseHeight': raw_data = getCloudBaseHeight(source,data)
  if variable == 'cloudCeiling':    raw_data = getCloudCeiling(source,data)

  raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

  # Array met_data is passed to MET
  # Graphics should plot $met_data to make sure things look correct
  if griddedDatasets[source]['flipY']:
     print('flipping ',source,' data about y-axis')
     met_data=np.flip(raw_data,axis=0).astype(float)
  else:
     met_data=raw_data.astype(float)

  # Make plotting optional or Just use plot_data_plane
#   plt_data=np.where(met_data<0, np.nan, met_data)
#   map=Basemap(projection='cyl',llcrnrlat=-90,urcrnrlat=90,llcrnrlon=-180,urcrnrlon=180,
↪resolution='c')
#   map.drawcoastlines()
#   map.drawcountries()
#   map.drawparallels(np.arange(-90,90,30),labels=[1,1,0,1])
#   map.drawmeridians(np.arange(0,360,60),labels=[1,1,0,1])
#   plt.contourf(lons,lats,plt_data,20,origin='upper',cmap=cm.Greens) #cm.gist_rainbow)
#   title=source+"_"+variable+"_"+str(validTime)
#   plt.title(title)
#   plt.colorbar(orientation='horizontal')
#   plt.savefig(title+".png")

  # If a forecast file, output a GRIB file with
  # 1 record containing the met_data
  # This is a hack, because right now, MET python embedding doesn't work with pygrib,
  #    so output the data to a temporary file, and then have MET read the temporary grib␣
↪file.
```

(continues on next page)

(continued from previous page)

```
   # Starting with version 9.0 of MET, the hack isn't needed, and MET python embedding works␣
↪with pygrib
   outputFcstFile = False  # MUST be True for MET version < 9.0.  For MET 9.0+, optional
   if dataSource == 1 and ftype == 'grib':
      if outputFcstFile:
         grbtmp = x
         grbtmp['values']=met_data
         grbout = open('temp_fcst.grb2','ab')
         grbout.write(grbtmp.tostring())
         grbout.close() # Close the outfile GRIB file
         print('Successfully output temp_fcst.grb2')

   # Close files
   if ftype == 'grib': idx.close()    # Close the input GRIB file
   if ftype == 'nc':   nc_fid.close() # Close the netCDF file

   return met_data

def obsError(fcstData,obsErrorFile,validDate,dataSource):

   print('Adding noise to the cloud fraction fields')
   print('Using obsErrorFile',obsErrorFile)

   # First load the obsError information
   #obsErrorFile = 'ob_errors.pk'
   infile = open(obsErrorFile,'rb')
   binEdges, binStddev = pk.load(infile) # 'numpy.ndarray' types
   infile.close()

   # Get 1d forecast data
   shape = fcstData.shape
   fcst = fcstData.flatten()

   # Set random number seed based on valid time and model
   if   dataSource.upper().strip() == 'MPAS':   ii = 10
   elif dataSource.upper().strip() == 'GALWEM': ii = 20
   elif dataSource.upper().strip() == 'GFS':    ii = 30
   np.random.seed(int(validDate*.1 + ii))

   # Find which bin the data is in
   for i in range(0,len(binEdges)-1):
      idx = np.where( (fcst >= binEdges[i]) & (fcst < binEdges[i+1]) )[0]
      n = len(idx) # number of points in the ith bin
      if n > 0: # check for empty bins
         randVals = np.random.normal(0,binStddev[i],n)
```

(continues on next page)

(continued from previous page)

```
        fcst[idx] = fcst[idx] + randVals

   # bound forecast values to between 0 and 100%
   fcst = np.where( fcst < 0.0,     0.0,   fcst)
   fcst = np.where( fcst > 100.0,   100.0, fcst)

   # now reshape forecast data back to 2D
   output = fcst.reshape(shape)

   # data will have NaNs where bad.
   return output

def getFcstCloudFrac(cfr,pmid,psfc,layerDefinitions): # cfr is cloud fraction(%), pmid is 3D␣
↪pressure(Pa), psfc is surface pressure (Pa) code from UPP ./INITPOST.F

   if pmid.shape != cfr.shape:  # sanity check
      print('dimension mismatch bewteen cldfra and pressure')
      sys.exit()

   nlocs, nlevs = pmid.shape

   if len(psfc) != nlocs: # another sanity check
      print('dimension mismatch bewteen cldfra and surface pressure')
      sys.exit()

   cfracl = np.zeros(nlocs)
   cfracm = np.zeros(nlocs)
   cfrach = np.zeros(nlocs)

   for i in range(0,nlocs):

      PTOP_HIGH = PTOP_HIGH_UPP
      if layerDefinitions.upper().strip() == 'ERA5':
         PTOP_LOW = 0.8*psfc[i]
         PTOP_MID = 0.45*psfc[i]
      elif layerDefinitions.upper().strip() == 'UPP':
         PTOP_LOW = PTOP_LOW_UPP
         PTOP_MID = PTOP_MID_UPP

      idxLow  = np.where(   pmid[i,:] >= PTOP_LOW)[0] # using np.where with just 1 argument␣
↪returns tuple
      idxMid  = np.where(  (pmid[i,:] <  PTOP_LOW) & (pmid[i,:] >= PTOP_MID))[0]
      idxHigh = np.where(  (pmid[i,:] <  PTOP_MID) & (pmid[i,:] >= PTOP_HIGH))[0]

      # use conditions in case all indices are missing
```

(continues on next page)

(continued from previous page)

```
      if (len(idxLow) >0 ):  cfracl[i] = np.max( cfr[i,idxLow] )
      if (len(idxMid) >0 ):  cfracm[i] = np.max( cfr[i,idxMid] )
      if (len(idxHigh) >0 ): cfrach[i] = np.max( cfr[i,idxHigh] )

   tmp = np.vstack( (cfracl,cfracm,cfrach)) # stack the rows into one 2d array
   cldfraMax = np.max(tmp,axis=0) # get maximum value across low/mid/high for each pixel␣
↪(minimum overlap assumption)

   # This is the fortran code put into python format...double loop unnecessary and slow
   #for i in range(0,nlocs):
   #   for k in range(0,nlevs):
   #      if pmid(i,k) >= PTOP_LOW:
   #         cfracl(i) = np.max( [cfracl(i),cfr(i,k)] ) # Low
   #      elif pmid(i,k) < PTOP_LOW and pmid(i,k) >= PTOP_MID:
   #         cfracm(i) = np.max( [cfracm(i),cfr(i,k)] ) # Mid
   #      elif pmid(i,k) < PTOP_MID and pmid(i,k) >= PTOP_HIGH: # High
   #         cfrach(i) = np.max( [cfrach(i),cfr(i,k)] )

   return cfracl, cfracm, cfrach, cldfraMax

def getGOES16LatLon(g16_data_file):

   # Start timer
   startTime = dt.datetime.utcnow()

   # designate dataset
   g16nc = Dataset(g16_data_file, 'r')

   # GOES-R projection info and retrieving relevant constants
   proj_info = g16nc.variables['goes_imager_projection']
   lon_origin = proj_info.longitude_of_projection_origin
   H = proj_info.perspective_point_height+proj_info.semi_major_axis
   r_eq = proj_info.semi_major_axis
   r_pol = proj_info.semi_minor_axis

   # Data info
   lat_rad_1d = g16nc.variables['x'][:]
   lon_rad_1d = g16nc.variables['y'][:]

   # close file when finished
   g16nc.close()
   g16nc = None

   # create meshgrid filled with radian angles
   lat_rad,lon_rad = np.meshgrid(lat_rad_1d,lon_rad_1d)
```

(continues on next page)

(continued from previous page)

```
   # lat/lon calc routine from satellite radian angle vectors

   lambda_0 = (lon_origin*np.pi)/180.0

   a_var = np.power(np.sin(lat_rad),2.0) + (np.power(np.cos(lat_rad),2.0)*(np.power(np.
↪cos(lon_rad),2.0)+(((r_eq*r_eq)/(r_pol*r_pol))*np.power(np.sin(lon_rad),2.0))))
   b_var = -2.0*H*np.cos(lat_rad)*np.cos(lon_rad)
   c_var = (H**2.0)-(r_eq**2.0)

   r_s = (-1.0*b_var - np.sqrt((b_var**2)-(4.0*a_var*c_var)))/(2.0*a_var)

   s_x = r_s*np.cos(lat_rad)*np.cos(lon_rad)
   s_y = - r_s*np.sin(lat_rad)
   s_z = r_s*np.cos(lat_rad)*np.sin(lon_rad)

   lat = (180.0/np.pi)*(np.arctan(((r_eq*r_eq)/(r_pol*r_pol))*((s_z/np.sqrt(((H-s_x)*(H-s_
↪x))+(s_y*s_y))))))
   lon = (lambda_0 - np.arctan(s_y/(H-s_x)))*(180.0/np.pi)

   # End timer
   endTime = dt.datetime.utcnow()
   time = (endTime - startTime).microseconds / (1000.0*1000.0)
   print('took %f4.1 seconds to get GOES16 lat/lon'%(time))

   return lon,lat # lat/lon are 2-d arrays

# --
def getGOESRetrivalData(goesFile,goesVar):

   if not os.path.exists(goesFile):
      print(goesFile+' not there. exit')
      sys.exit()

   # First get GOES lat/lon
   goesLon2d, goesLat2d = getGOES16LatLon(goesFile) # 2-d arrays
   goesLon = goesLon2d.flatten() # 1-d arrays
   goesLat = goesLat2d.flatten()

   # Now open the file and get the data we want
   nc_goes = Dataset(goesFile, "r", format="NETCDF4")

   # If the next line is true (it should be), this indicates the variable needs to be treated
   #  as an "unsigned 16-bit integer". This is a pain.  So we must use the "astype" method
   #  to change the variable type BEFORE applying scale_factor and add_offset.  After the␣
```

(continues on next page)

(continued from previous page)

```
↪conversion
   #  we then can manually apply the scale factor and offset
   #goesVar = 'PRES'
   goesVar = goesVar.strip() # for safety
   if nc_goes.variables[goesVar]._Unsigned.lower().strip() == 'true':
      nc_goes.set_auto_scale(False) # Don't automatically apply scale_factor and add_offset␣
↪to variable
      goesData2d = np.array( nc_goes.variables[goesVar]).astype(np.uint16)
      goesData2d = goesData2d * nc_goes.variables[goesVar].scale_factor + nc_goes.
↪variables[goesVar].add_offset
      goesQC2d  = np.array( nc_goes.variables['DQF']).astype(np.uint8)
   else:
      goesData2d = np.array( nc_goes.variables[goesVar])
      goesQC2d  = np.array( nc_goes.variables['DQF'])

   # Make variables 1-d
   goesQC  = goesQC2d.flatten()
   goesData = goesData2d.flatten()
   nc_goes.close()

   # Get rid of NaNs; base it on longitude
   goesData = goesData[~np.isnan(goesLon)] # Handle data arrays first before changing lat/
↪lon itself
   goesQC  = goesQC[~np.isnan(goesLon)]
   goesLon = goesLon[~np.isnan(goesLon)] # ~ is "logical not", also np.logical_not
   goesLat = goesLat[~np.isnan(goesLat)]
   if goesLon.shape != goesLat.shape:
      print('GOES lat/lon shape mismatch')
      sys.exit()

   # If goesQC == 0, good QC and there was a cloud with a valid pressure.
   # If goesQC == 4, no cloud; probably clear sky.
   # All other QC means no data, and we want to remove those points
   idx = np.logical_or( goesQC == 0, goesQC == 4) # Only keep QC == 0 or 4
   goesData = goesData[idx]
   goesQC  = goesQC[idx]
   goesLon = goesLon[idx]
   goesLat = goesLat[idx]

   # Only QC with 0 or 4 are left; now set QC == 4 to missing to indicate clear sky
   goesData = np.where( goesQC != 0, missing_values, goesData)

   # Get longitude to between (0,360) for consistency with JEDI files (this check is applied␣
↪to JEDI files, too)
   goesLon = np.where( goesLon < 0, goesLon + 360.0, goesLon )
```

(continues on next page)

(continued from previous page)

```
   print('Min GOES Lon = ',np.min(goesLon))
   print('Max GOES Lon = ',np.max(goesLon))

   return goesLon, goesLat, goesData

def point2point(source,inputDir,satellite,channel,goesFile,condition,layerDefinitions,
↪dataSource):

   # Static Variables for QC and obs
   qcVar  = 'brightness_temperature_'+str(channel)+'@EffectiveQC' #'@EffectiveQC0' # QC␣
↪variable
   obsVar = 'brightness_temperature_'+str(channel)+'@ObsValue'  # Observation variable

   # Get GOES-16 retrieval file with auxiliary information
   if 'abi' in satellite or 'ahi' in satellite:
      goesLon, goesLat, goesData = getGOESRetrivalData(goesFile,'PRES') # return 1-d arrays
      lonlatGOES = np.array( list(zip(goesLon, goesLat))) # lon/lat pairs for each GOES ob␣
↪(nobs_GOES, 2)
     #print('shape lonlatGOES = ',lonlatGOES.shape)
      print('getting data from ',goesFile)
      myGOESInterpolator = NearestNDInterpolator(lonlatGOES,goesData)

   # First check to see if there's a concatenated file with all obs.
   #  If so, use that.  If not, have to process one file per processor, which takes a lot␣
↪more time
   if os.path.exists(inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'):
      inputFiles =  [inputDir+'/obsout_omb_'+satellite+'_ALL.nc4'] # needs to be in a list␣
↪since we loop over inputFiles
   else:
      # Get list of OMB files to process.  There is one file per processor.
      # Need to get them in order so they are called in the same order for the
      # forecast and observed passes through this subroutine.
      files = os.listdir(inputDir)
      inputFiles = fnmatch.filter(files,'obsout*_'+satellite+'*nc4') # returns relative path␣
↪names
      inputFiles = [inputDir+'/'+s for s in inputFiles] # add on directory name
      inputFiles.sort() # Get in order from low to high
   if len(inputFiles) == 0: return -99999, -99999 # if no matching files, force a failure

   # Variable to pull for brightness temperature
#  if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@GsiHofXBc' # Forecast␣
↪variable
   if dataSource == 1: v = 'brightness_temperature_'+str(channel)+'@hofx' #'@depbg' # OMB
   if dataSource == 2: v = obsVar
```

(continues on next page)

(continued from previous page)

```
  # Read the files and put data in array
  allData, allDataQC = [], []
  for inputFile in inputFiles:
     nc_fid = Dataset(inputFile, "r", format="NETCDF4") #Dataset is the class behavior to␣
↪open the file
     print('Trying to read ',v,' from ',inputFile)

     # Read forecast/obs data
     read_var = nc_fid.variables[v]         # extract/copy the data
  #  read_missing = read_var.missing_value  # get variable attributes. Each dataset has own␣
↪missing values.
     this_var = np.array( read_var )        # to numpy array
  #  this_var = np.where( this_var==read_missing, np.nan, this_var )

    #if dataSource == 1: # If true, we just read in OMB data, but we want B
    #   obsData = np.array( nc_fid.variables[obsVar])
    #   this_var = obsData - this_var # get background/forecast value (O - OMB = B)

     #Read QC data
     qcData = np.array(nc_fid.variables[qcVar])

     # Sanity check...shapes should match
     if qcData.shape != this_var.shape: return -99999, -99999

     if 'abi' in satellite or 'ahi' in satellite:

        # Get the GOES-16 retrieval data at the observation locations in this file
        #   GOES values < 0 mean clear sky
        lats = np.array(nc_fid.variables['latitude@MetaData'])
        lons = np.array(nc_fid.variables['longitude@MetaData'])

        # Get longitude to between (0,360) for consistency with GOES-16 files
        lons = np.where( lons < 0, lons + 360.0, lons )

        lonlat = np.array( list(zip(lons,lats)))  # lon/lat pairs for each ob (nobs, 2)
        thisGOESData = myGOESInterpolator(lonlat) # GOES data at obs locations in this file.
↪ If pressure, units are hPa
        thisGOESData = thisGOESData * 100.0 # get into Pa

        #obsCldfra = np.array( nc_fid.variables['cloud_area_fraction@MetaData'] )*100.0 #␣
↪Get into %...observed cloud fraction (AHI/ABI only)

        geoValsFile = inputFile.replace('obsout','geoval')
        if not os.path.exists(geoValsFile):
```

(continues on next page)

(continued from previous page)

```
            print(geoValsFile+' not there. exit')
            sys.exit()

         nc_fid2 = Dataset(geoValsFile, "r", format="NETCDF4")
         fcstCldfra = np.array( nc_fid2.variables['cloud_area_fraction_in_atmosphere_layer
↪'])*100.0 # Get into %
         pressure   = np.array( nc_fid2.variables['air_pressure']) # Pa
         pressure_edges   = np.array( nc_fid2.variables['air_pressure_levels']) # Pa
         psfc = pressure_edges[:,-1]  # Surface pressure (Pa)...array order is top down
         if layerDefinitions.upper().strip() == 'ERA5':
            PTOP_LOW = 0.8*psfc # these are arrays
            PTOP_MID = 0.45*psfc
            PTOP_HIGH = PTOP_HIGH_UPP * np.ones_like(psfc)
         elif layerDefinitions.upper().strip() == 'UPP':
            PTOP_LOW = PTOP_LOW_UPP # these are constants
            PTOP_MID = PTOP_MID_UPP
            PTOP_HIGH = PTOP_HIGH_UPP
         else:
            print('layerDefinitions = ',layerDefinitions,'is invalid. exit')
            sys.exit()
         fcstLow,fcstMid,fcstHigh,fcstTotCldFra = getFcstCloudFrac(fcstCldfra,pressure,psfc,
↪layerDefinitions) # get low/mid/high/total forecast cloud fractions for each ob
         nc_fid2.close()

         # Modify QC data based on correspondence between forecast and obs. qcData used to␣
↪select good data later
         # It's possible that there are multiple forecast layers, such that fcstLow,fcstMid,
↪fcstHigh are all > $cldfraThresh
         # However, GOES-16 CTP doesn't really account for layering.  So, we need to remove␣
↪layered clouds from the forecast,
         #   focusing only on the layers that we asked for when doing {low,mid,high}Only␣
↪conditions
         # The "|" is symbol for "np.logcal_or"
         yes = 2.0
         no  = 0.0
         cldfraThresh = 20.0 # percent
         if qcData.shape == fcstTotCldFra.shape == thisGOESData.shape:  # these should all␣
↪match
            print('Using condition ',condition,'for ABI/AHI')

            # Note that "&" is "np.logical_and" for boolean (true/false) quantities.
            # Thus, each condition should be enclosed in parentheses
            if   condition.lower().strip() == 'clearOnly'.lower():  # clear in both forecast␣
↪and obs
               qcData = np.where( (fcstTotCldFra < cldfraThresh)  & (thisGOESData <= 0.0),␣
```

(continues on next page)

(continued from previous page)

```
↪qcData, missing_values)
            elif condition.lower().strip() == 'cloudyOnly'.lower(): # cloudy in both␣
↪forecast and obs
               qcData = np.where( (fcstTotCldFra >= cldfraThresh) & (thisGOESData > 0.0),␣
↪qcData, missing_values)
            elif condition.lower().strip() == 'lowOnly'.lower(): # low clouds in both␣
↪forecast and obs
               fcstLow = np.where( (fcstMid >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstLow) # remove mid, high
               qcData = np.where( (fcstLow >= cldfraThresh) & ( thisGOESData >= PTOP_LOW),␣
↪qcData, missing_values)
            elif condition.lower().strip() == 'midOnly'.lower(): # mid clouds in both␣
↪forecast and obs
               fcstMid = np.where( (fcstLow >= cldfraThresh) | ( fcstHigh >= cldfraThresh),␣
↪missing_values, fcstMid) # remove low, high
               qcData = np.where( (fcstMid >= cldfraThresh) & (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID),   qcData, missing_values)
            elif condition.lower().strip() == 'highOnly'.lower(): # high clouds in both␣
↪forecast and obs
               fcstHigh = np.where( (fcstLow >= cldfraThresh) | ( fcstMid >= cldfraThresh),␣
↪missing_values, fcstHigh) # remove mid, high
               qcData = np.where( (fcstHigh >= cldfraThresh) & (thisGOESData <  PTOP_MID) &␣
↪(thisGOESData >= PTOP_HIGH), qcData, missing_values)
            elif condition.lower().strip() == 'fcstLow'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
               qcData = np.where( fcstLow >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'fcstMid'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
               qcData = np.where( fcstMid >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'fcstHigh'.lower(): # low clouds in forecast␣
↪(layers possible); obs could be anything
               qcData = np.where( fcstHigh >= cldfraThresh , qcData, missing_values)
            elif condition.lower().strip() == 'cloudEventLow'.lower():
               if dataSource == 1: this_var = np.where( fcstLow      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
               if dataSource == 2: this_var = np.where( thisGOESData >= PTOP_LOW, yes, no )
            elif condition.lower().strip() == 'cloudEventMid'.lower():
               if dataSource == 1: this_var = np.where( fcstMid      >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
               if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_LOW) &␣
↪(thisGOESData >= PTOP_MID), yes, no )
            elif condition.lower().strip() == 'cloudEventHigh'.lower():
               if dataSource == 1: this_var = np.where( fcstHigh     >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
               if dataSource == 2: this_var = np.where( (thisGOESData <  PTOP_MID) &␣
```

(continues on next page)

(continued from previous page)

```
↪(thisGOESData >= PTOP_HIGH), yes, no )
            elif condition.lower().strip() == 'cloudEventTot'.lower():
               if dataSource == 1: this_var = np.where( fcstTotCldFra >= cldfraThresh, yes,␣
↪no ) # set cloudy points to 2, clear points to 0, use threshold of 1 in MET
               if dataSource == 2: this_var = np.where( thisGOESData  > 0.0, yes, no )
            elif condition.lower().strip() == 'all':
               print("not doing any conditional verification or stratifying by event")
            else:
               print("condition = ",condition," not recognized.")
               sys.exit()
            #elif condition.lower().strip() == '4x4table'.lower():
              #if dataSource == 1:
              #   this_var = np.where( fcstLow >= cldfraThresh, yesLow, no )
              #   this_var = this_var + np.where( fcstMid >= cldfraThresh, yesMid, no )
              #   this_var = this_var + np.where( fcstHigh >= cldfraThresh, yesHigh, no )
            print('number removed = ', (qcData==missing_values).sum())
           #print('number passed   = ', qcData.shape[0] - (qcData==missing_values).sum())
         else:
            print('shape mismatch')
            return -99999, -99999

      # Append to arrays
      allData.append(this_var)
      allDataQC.append(qcData)

      nc_fid.close() # done with the file, so close it before going to next file in loop

   # We're now all done looping over the individul files

   # Get the indices with acceptable QC
   allQC = np.concatenate(allDataQC) # Put list of numpy arrays into a single long 1-D numpy␣
↪array.  All QC data.
   idx = np.where(allQC==0) # returns indices

   # Now get all the forecast/observed brightness temperature data with acceptable QC
   this_var = np.concatenate(allData)[idx] # Put list of numpy arrays into a single long 1-D␣
↪numpy array. This is all the forecast/obs data with good QC
   numObs = this_var.shape[0] # number of points with good QC for this channel
   print('Number of obs :',numObs)

   # Assume all the points actually fit into a square grid. Get the side of the square (use␣
↪ceil to round up)
   if numObs > 0:
      l = np.ceil(np.sqrt(numObs)).astype('int') # Length of the side of the square
```

(continues on next page)

(continued from previous page)

```
      # Make an array that can be reshaped into the square
      raw_data1D = np.full(l*l,np.nan) # Initialize 1D array of length l**2 to np.nan
      raw_data1D[0:numObs] = this_var[:] # Fill data to the extent possible. There will be␣
↪some np.nan values at the end
      raw_data = np.reshape(raw_data1D,(l,l)) # Reshape into "square grid"

      raw_data = np.where(np.isnan(raw_data), missing_values, raw_data) # replace np.nan to␣
↪missing_values (for MET)

      met_data=raw_data.astype(float) # Give MET this info

      # Now need to tell MET the "grid" for the data
      # Make a fake lat/lon grid going from 0.0 to 50.0 degrees, with the interval␣
↪determined by number of points
      griddedDatasets[source]['latDef'][0] = 0.0 # starting point
      griddedDatasets[source]['latDef'][1] = np.diff(np.linspace(0,50,l)).round(6)[0] #␣
↪interval (degrees)
      griddedDatasets[source]['latDef'][2] = int(l) # number of points
      griddedDatasets[source]['lonDef'][0:3] = griddedDatasets[source]['latDef']

      gridInfo = getGridInfo(source, griddedDatasets[source]['gridType']) # 'LatLon' gridType
      return met_data, gridInfo

   else:
      return -99999, -99999

###########
def getGridInfo(source,gridType):

   if gridType == 'LatLon':
      latDef = griddedDatasets[source]['latDef']
      lonDef = griddedDatasets[source]['lonDef']
      gridInfo = {
         'type':      gridType,
         'name':      source,
         'lat_ll':    latDef[0], #-90.000,
         'lon_ll':    lonDef[0], #-180.000,
         'delta_lat': latDef[1], #0.5000,
         'delta_lon': lonDef[1], #0.625,
         'Nlat':      latDef[2], #361,
         'Nlon':      lonDef[2], #576,
      }
   elif gridType == 'Gaussian':
      gridInfo = {
        'type':     gridType,
```

(continues on next page)

(continued from previous page)

```
        'name':     source,
        'nx':       griddedDatasets[source]['nx'],
        'ny':       griddedDatasets[source]['ny'],
        'lon_zero': griddedDatasets[source]['lon_zero'],
      }

   return gridInfo

def getAttrArray(source,variable,initTime,validTime):

   init = dt.datetime.strptime(initTime,"%Y%m%d%H")
   valid = dt.datetime.strptime(validTime,"%Y%m%d%H")
   lead, rem = divmod((valid-init).total_seconds(), 3600)

   attrs = {

      'valid': valid.strftime("%Y%m%d_%H%M%S"),
      'init':  init.strftime("%Y%m%d_%H%M%S"),
      'lead':  str(int(lead)),
      'accum': '000000',

      'name':      variable,  #'MERRA2_Cloud_Percentage'
      'long_name': variable,  #'Cloud Percentage Levels',
      'level':     'ALL',
      'units':     verifVariables[variable]['units'],

      'grid': getGridInfo(source,griddedDatasets[source]['gridType'])
   }

   #print(attrs)
   #print(griddedDatasets[source])

   return attrs

######## END FUNCTIONS   ##########


#if __name__ == "__main__":
dataFile, dataSource, variable, i_date, v_date, flag = sys.argv[1].split(":")
met_data = getDataArray(dataFile,dataSource,variable,flag)
attrs = getAttrArray(dataSource,variable,i_date,v_date)
print(attrs)
```

Running METplus

Pass the use case configuration file to the run_metplus.py script along with any user-specific system configuration files if desired:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/clouds/GridStat_fcstMPAS_
↪obsMERRA2_lowAndTotalCloudFrac.conf /path/to/user_system.conf
```

See *Running METplus* (page 26) for more information.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/clouds/GridStat_fcstMPAS_obsMERRA2_lowAndTotalCloudFrac (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_MPAS_to_MERRA2_F36_CloudFracs_360000L_20200724_120000V_pairs.nc
- grid_stat_MPAS_to_MERRA2_F36_CloudFracs_360000L_20200724_120000V.stat
- grid_stat_MPAS_to_MERRA2_F36_CloudFracs_NBR_360000L_20200724_120000V_pairs.nc
- grid_stat_MPAS_to_MERRA2_F36_CloudFracs_NBR_360000L_20200724_120000V.stat
- grid_stat_MPAS_to_MERRA2_F36_CloudFracs_PROB_360000L_20200724_120000V_pairs.nc
- grid_stat_MPAS_to_MERRA2_F36_CloudFracs_PROB_360000L_20200724_120000V.stat

Keywords

Note:

- GridStatToolUseCase
- NetCDFFileUseCase
- CloudsAppUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/clouds-GridStat_fcstMPAS_obsMERRA2_lowAndTotalCloudFrac.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.4 Data Assimilation

Observational data used as part of the initial conditions for numerical weather prediction

7.2.16.4.1 StatAnalysis: IODAv1

model_applications/data_assimilation/StatAnalysis_fcstHAFS_obsPrepBufr_JEDI_IODA_interface.conf

Scientific Objective

This use case demonstrates the Stat-Analysis tool and ingestion of HofX netCDF files that have been output from the Joint Effort for Data assimilation Integration (JEDI) data assimilation system. JEDI uses "IODA" formatted files, which are netCDF files with certain requirements of variables and naming conventions. These files hold observations to be assimilated into forecasts, in this case the FV3-based Hurricane Analysis and Forecast System (HAFS). HAFS performs tc initialization by using synthetic observations of conventional variables to relocate a tropical cyclone as informed by a vortex tracker, in this case Tropical Storm Dorian.

In this case 100224 observations from 2019082418 are used. These were converted from perpbufr files via a fortran ioda-converter provided by the Joint Center for Satellite Data Assimilation, which oversees the development of JEDI. The variables used are t, q, u, and v.

The first component of JEDI to be incorporated into operational systems will be the Unified Forward Operator (UFO) to replace the GSI observer in global EnKF forecasts. UFO is a component of HofX, which maps the background forecast to observation space to form O minus B pairs. The HofX application of JEDI takes the input IODA files and adds an additional variable, <variable_name>@hofx that is to be paired with <variable_name>@ObsValue. These HofX files are used as input to form Matched Pair (MPR) formatted lists via Python embedding. In this case, Stat-Analysis then performs a filter job and outputs the filtered MPR formatted columns in an ascii file.

Datasets

Data source: JEDI HofX output files in IODA format

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1032) section for more information.

METplus Components

This use case utilizes the METplus StatAnalysis wrapper to search for files that are valid for the given case and generate a command to run the MET tool stat_analysis.

METplus Workflow

StatAnalysis is the only tool called in this example. It processes the following run times:

Valid: 2019-08-24_18Z
Forecast lead: 6 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/data_assimilation/StatAnalysis_fcstHAFS_obsPrepBufr_JEDI_IODA_interface.con

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/data_assimilation/
↪StatAnalysis_fcstHAFS_obsPrepBufr_JEDI_IODA_interface.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = StatAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID

VALID_TIME_FMT = %Y%m%d%H
VALID_BEG=2005080700
VALID_END=2005080700
VALID_INCREMENT = 12H

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = python {PARM_BASE}/use_cases/model_applications/data_
↪assimilation/StatAnalysis_fcstHAFS_obsPrepBufr_JEDI_IODA_interface/read_ioda_mpr.py {INPUT_
↪BASE}/model_applications/data_assimilation/hofx_dir

STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/data_assimilation/StatAnalysis_
↪HofX

MODEL1_STAT_ANALYSIS_DUMP_ROW_TEMPLATE = dump.out


###
# StatAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#statanalysis
###

MODEL1 = NA
MODEL1_OBTYPE = NA

STAT_ANALYSIS_JOB_NAME = filter
STAT_ANALYSIS_JOB_ARGS = -out_line_type CNT -dump_row [dump_row_file] -line_type MPR

MODEL_LIST =
DESC_LIST =
FCST_LEAD_LIST =
```

(continues on next page)

(continued from previous page)

```
OBS_LEAD_LIST =
FCST_VALID_HOUR_LIST =
FCST_INIT_HOUR_LIST =
OBS_VALID_HOUR_LIST =
OBS_INIT_HOUR_LIST =
FCST_VAR_LIST =
OBS_VAR_LIST =
FCST_UNITS_LIST =
OBS_UNITS_LIST =
FCST_LEVEL_LIST =
OBS_LEVEL_LIST =
VX_MASK_LIST =
INTERP_MTHD_LIST =
INTERP_PNTS_LIST =
FCST_THRESH_LIST =
OBS_THRESH_LIST =
COV_THRESH_LIST =
ALPHA_LIST =
LINE_TYPE_LIST =

GROUP_LIST_ITEMS =
LOOP_LIST_ITEMS = MODEL_LIST
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *StatAnalysis MET Configuration* (page 246) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// STAT-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Filtering input STAT lines by the contents of each column
//
${METPLUS_MODEL}
${METPLUS_DESC}

${METPLUS_FCST_LEAD}
${METPLUS_OBS_LEAD}

${METPLUS_FCST_VALID_BEG}
${METPLUS_FCST_VALID_END}
${METPLUS_FCST_VALID_HOUR}

${METPLUS_OBS_VALID_BEG}
${METPLUS_OBS_VALID_END}
${METPLUS_OBS_VALID_HOUR}

${METPLUS_FCST_INIT_BEG}
${METPLUS_FCST_INIT_END}
${METPLUS_FCST_INIT_HOUR}

${METPLUS_OBS_INIT_BEG}
${METPLUS_OBS_INIT_END}
${METPLUS_OBS_INIT_HOUR}

${METPLUS_FCST_VAR}
${METPLUS_OBS_VAR}

${METPLUS_FCST_UNITS}
${METPLUS_OBS_UNITS}

${METPLUS_FCST_LEVEL}
${METPLUS_OBS_LEVEL}

${METPLUS_OBTYPE}

${METPLUS_VX_MASK}

${METPLUS_INTERP_MTHD}

${METPLUS_INTERP_PNTS}

${METPLUS_FCST_THRESH}
${METPLUS_OBS_THRESH}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_COV_THRESH}

${METPLUS_ALPHA}

${METPLUS_LINE_TYPE}

column = [];

weight = [];

////////////////////////////////////////////////////////////////////////////////

//
// Array of STAT-Analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
out_alpha = 0.05;

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// WMO mean computation logic
//
wmo_sqrt_stats   = [ "CNT:FSTDEV",  "CNT:OSTDEV",  "CNT:ESTDEV",
                     "CNT:RMSE",    "CNT:RMSFA",   "CNT:RMSOA",
                     "VCNT:FS_RMS", "VCNT:OS_RMS", "VCNT:RMSVE",
                     "VCNT:FSTDEV", "VCNT:OSTDEV" ];

wmo_fisher_stats = [ "CNT:PR_CORR", "CNT:SP_CORR",
                     "CNT:KT_CORR", "CNT:ANOM_CORR" ];
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;
vif_flag       = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses a Python embedding script to read input data

parm/use_cases/model_applications/data_assimilation/StatAnalysis_fcstHAFS_obsPrepBufr_JEDI_IODA_interface/rea

```
from __future__ import print_function

import pandas as pd
import os
from glob import glob
import sys
import xarray as xr
import datetime as dt

################################################################################

def read_netcdfs(files, dim):
    paths = sorted(glob(files))
    datasets = [xr.open_dataset(p) for p in paths]
    combined = xr.concat(datasets, dim)
    return combined

################################################################################
print('Python Script:\t', sys.argv[0])

# Input is directory of .nc or .nc4 files

if len(sys.argv) == 2:
    # Read the input file as the first argument
    input_dir = os.path.expandvars(sys.argv[1])
    try:
```

(continues on next page)

(continued from previous page)

```
        print("Input File:\t" + repr(input_dir))

        # Read all from a directory
        ioda_data = read_netcdfs(input_dir+'/*.nc*', dim='nlocs')

        # Grab variables list
        var_list = ioda_data['variable_names@VarMetaData'].isel(nlocs=[0]).str.decode('utf-8
↪').values
        var_list = [i.strip() for i in var_list[0] if i]

        # Use only nlocs dimension to ensure a table
        ioda_data = ioda_data.drop_dims('nvars')
        ioda_df = ioda_data.to_dataframe()

        nlocs = len(ioda_df.index)
        print('Number of locations in set: ' + str(nlocs))

        # Decode strings
        ioda_df.loc[:,'datetime@MetaData'] = ioda_df.loc[:,'datetime@MetaData'].str.decode(
↪'utf-8')
        ioda_df.loc[:,'station_id@MetaData'] = ioda_df.loc[:,'station_id@MetaData'].str.
↪decode('utf-8')

        # Datetime format. Need YYYYMMD_HHMMSS from YYYY-MM-DDTHH:MM:SSZ.
        time = ioda_df.loc[:,'datetime@MetaData'].values.tolist()

        for i in range(0,nlocs):
            temp = dt.datetime.strptime(time[i], '%Y-%m-%dT%H:%M:%SZ')
            time[i] = temp.strftime('%Y%m%d_%H%M%S')

        ioda_df.loc[:,'datetime@MetaData'] = time

        mpr_data = []
        var_list = [i for i in var_list if i+'@hofx' in ioda_df.columns]

        for var_name in var_list:

            # Subset the needed columns
            ioda_df_var = ioda_df[['datetime@MetaData','station_id@MetaData',var_name+
↪'@ObsType',
                                   'latitude@MetaData','longitude@MetaData','air_
↪pressure@MetaData',
                                   var_name+'@hofx',var_name+'@ObsValue',
                                   var_name+'@PreQC']]
```

(continues on next page)

(continued from previous page)

```
            # Find locations with ObsValues
            ioda_df_var = ioda_df_var[ioda_df_var[var_name+'@ObsValue'] < 1e9]
            nlocs = len(ioda_df_var.index)
            print(var_name+' has '+str(nlocs)+' obs.')

            # Add additional columns
            ioda_df_var['lead'] = '000000'
            ioda_df_var['MPR'] = 'MPR'
            ioda_df_var['nobs'] = nlocs
            ioda_df_var['index'] = range(0,nlocs)
            ioda_df_var['varname'] = var_name
            ioda_df_var['na'] = 'NA'

            # Arrange columns in MPR format
            cols = ['na','na','lead','datetime@MetaData','datetime@MetaData','lead',
↪'datetime@MetaData',
                    'datetime@MetaData','varname','na','lead','varname','na','na',
                    var_name+'@ObsType','na','na','lead','na','na','na','na','MPR',
                    'nobs','index','station_id@MetaData','latitude@MetaData',
↪'longitude@MetaData',
                    'air_pressure@MetaData','na',var_name+'@hofx',var_name+'@ObsValue',
                    var_name+'@PreQC','na','na']

            ioda_df_var = ioda_df_var[cols]

            # Into a list and all to strings
            mpr_data = mpr_data + [list( map(str,i) ) for i in ioda_df_var.values.tolist() ]

            print("Total Length:\t" + repr(len(mpr_data)))

    except NameError:
        print("Can't find the input files or the variables.")
        print("Variables in this file:\t" + repr(var_list))
else:
    print("ERROR: read_ioda_mpr.py -> Must specify directory of files.\n")
    sys.exit(1)

##############################################################################
```

Running METplus

It is recommended to run this use case by:

Passing in StatAnalysis_fcstHAFS_obsPrepBufr_JEDI_IODA_interface.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/StatAnalysis_fcstHAFS_obsPrepBufr_JEDI_IODA_interface.conf -c /
↪path/to/user_system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/data_assimilation/StatAnalysis_HofX (relative to **OUTPUT_BASE**) and will contain the following file:

- dump.out

Keywords

Note:

- StatAnalysisToolUseCase
- PythonEmbeddingFileUseCase
- TCandExtraTCAppUseCase
- NOAAEMCOrgUseCase
- IODA2NCToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/data_assimilation-StatAnalysis_fcstHAFS_obsPrepBufr_JEDI_IODA_interface

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.4.2 StatAnalysis: IODAv2

model_applications/data_assimilation/StatAnalysis_fcstGFS_HofX_obsIODAv2_PyEmbed.conf

Scientific Objective

This use case demonstrates the Stat-Analysis tool and ingestion of HofX NetCDF files that have been output from the Joint Effort for Data assimilation Integration (JEDI) data assimilation system. JEDI uses IODA version 2 formatted files, which are NetCDF files with certain requirements of variables and naming conventions. These files hold observations to be assimilated into forecasts, in this case taken from the JEDI software test data, which contained a small number of Global observation-forecast pairs derived from the hofx application.

UFO is a component of HofX, which maps the background forecast to observation space to form O minus B pairs. The HofX application of JEDI takes the input IODAv2 files and adds an additional variable which is the forecast value as interpolated to the observation location. These HofX files are used as input to form Matched Pair (MPR) formatted lists via Python embedding. In this case, Stat-Analysis then performs an aggregate_stat job and outputs statistics in an ascii file.

This use case adopts the IODAv2 formatted NetCDF files, which replace the previous variable formatting scheme to make use of NetCDF groups.

Datasets

Data source: JEDI HofX output files in IODAv2 format

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1042) section for more information.

METplus Components

This use case utilizes the METplus StatAnalysis wrapper to search for files that are valid for the given case and generate a command to run the MET tool stat_analysis.

METplus Workflow

StatAnalysis is the only tool called in this example. It processes the following run times:

Valid: 2018-04-15_00Z
Forecast lead: 0 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/data_assimilation/StatAnalysis_fcstGFS_HofX_obsIODAv2_PyEmbed.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/data_assimilation/
↪StatAnalysis_fcstGFS_HofX_obsIODAv2_PyEmbed.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = StatAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID

VALID_TIME_FMT = %Y%m%d%H
VALID_BEG=2018041500
VALID_END=2018041500
VALID_INCREMENT = 12H

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = python {PARM_BASE}/use_cases/model_applications/data_
↪assimilation/StatAnalysis_fcstGFS_HofX_obsIODAv2_PyEmbed/read_iodav2_mpr.py {INPUT_BASE}/
↪model_applications/data_assimilation/StatAnalysis_fcstGFS_HofX_obsIODAv2_PyEmbed/sample_
↪hofx_output_sondes.nc4

STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/StatAnalysis_IODAv2
STAT_ANALYSIS_OUTPUT_TEMPLATE = job.out
MODEL1_STAT_ANALYSIS_DUMP_ROW_TEMPLATE = dump.out


###
```

(continues on next page)

(continued from previous page)

```
# StatAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#statanalysis
###

MODEL1 = NA
MODEL1_OBTYPE = NA

STAT_ANALYSIS_JOB_NAME = aggregate_stat
STAT_ANALYSIS_JOB_ARGS = -out_line_type CNT -dump_row [dump_row_file] -line_type MPR -by␣
↪FCST_VAR

MODEL_LIST =
DESC_LIST =
FCST_LEAD_LIST =
OBS_LEAD_LIST =
FCST_VALID_HOUR_LIST =
FCST_INIT_HOUR_LIST =
OBS_VALID_HOUR_LIST =
OBS_INIT_HOUR_LIST =
FCST_VAR_LIST =
OBS_VAR_LIST =
FCST_UNITS_LIST =
OBS_UNITS_LIST =
FCST_LEVEL_LIST =
OBS_LEVEL_LIST =
VX_MASK_LIST =
INTERP_MTHD_LIST =
INTERP_PNTS_LIST =
FCST_THRESH_LIST =
OBS_THRESH_LIST =
COV_THRESH_LIST =
ALPHA_LIST =
LINE_TYPE_LIST =

GROUP_LIST_ITEMS =
LOOP_LIST_ITEMS = MODEL_LIST
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *StatAnalysis MET Configuration* (page 246) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// STAT-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Filtering input STAT lines by the contents of each column
//
${METPLUS_MODEL}
${METPLUS_DESC}

${METPLUS_FCST_LEAD}
${METPLUS_OBS_LEAD}

${METPLUS_FCST_VALID_BEG}
${METPLUS_FCST_VALID_END}
${METPLUS_FCST_VALID_HOUR}

${METPLUS_OBS_VALID_BEG}
${METPLUS_OBS_VALID_END}
${METPLUS_OBS_VALID_HOUR}

${METPLUS_FCST_INIT_BEG}
${METPLUS_FCST_INIT_END}
${METPLUS_FCST_INIT_HOUR}

${METPLUS_OBS_INIT_BEG}
${METPLUS_OBS_INIT_END}
${METPLUS_OBS_INIT_HOUR}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_FCST_VAR}
${METPLUS_OBS_VAR}

${METPLUS_FCST_UNITS}
${METPLUS_OBS_UNITS}

${METPLUS_FCST_LEVEL}
${METPLUS_OBS_LEVEL}

${METPLUS_OBTYPE}

${METPLUS_VX_MASK}

${METPLUS_INTERP_MTHD}

${METPLUS_INTERP_PNTS}

${METPLUS_FCST_THRESH}
${METPLUS_OBS_THRESH}
${METPLUS_COV_THRESH}

${METPLUS_ALPHA}

${METPLUS_LINE_TYPE}

column = [];

weight = [];

////////////////////////////////////////////////////////////////////////////////

//
// Array of STAT-Analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
out_alpha = 0.05;

boot = {
```

(continues on next page)

(continued from previous page)

```
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// WMO mean computation logic
//
wmo_sqrt_stats   = [ "CNT:FSTDEV",  "CNT:OSTDEV",  "CNT:ESTDEV",
                     "CNT:RMSE",    "CNT:RMSFA",   "CNT:RMSOA",
                     "VCNT:FS_RMS", "VCNT:OS_RMS", "VCNT:RMSVE",
                     "VCNT:FSTDEV", "VCNT:OSTDEV" ];

wmo_fisher_stats = [ "CNT:PR_CORR", "CNT:SP_CORR",
                     "CNT:KT_CORR", "CNT:ANOM_CORR" ];

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;
vif_flag       = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses a Python embedding script to read input data

parm/use_cases/model_applications/data_assimilation/StatAnalysis_fcstGFS_HofX_obsIODAv2_PyEmbed/read_iodav

```
from __future__ import print_function

import pandas as pd
import os
from glob import glob
import sys
```

(continues on next page)

(continued from previous page)

```
import xarray as xr
import datetime as dt

########################################################################

print('Python Script:\t', sys.argv[0])

# Input is .nc or .nc4 file

if len(sys.argv) == 2:
    # Read the input file as the first argument
    input_path = os.path.expandvars(sys.argv[1])
    try:
        print("Input File:\t" + repr(input_path))

        # Read all the needed groups
        ioda_data = xr.open_dataset(input_path, group = 'MetaData')
        ioda_hofx_data = xr.open_dataset(input_path, group = 'hofx')

        hofx_vars = list(ioda_hofx_data.keys())

        # use dataframes
        ioda_df = ioda_data.to_dataframe()
        ioda_data.close()

        for var_name in hofx_vars:
            ioda_df[var_name + '@hofx'] = ioda_hofx_data[var_name]

        # Add columns for needed attributes, for each variable present for hofx
        for attribute in ['ObsValue', 'ObsType', 'EffectiveQC']:
            ioda_attr_data = xr.open_dataset(input_path, group = attribute)
            for var_name in hofx_vars:
                ioda_df[var_name + '@' + attribute] = ioda_attr_data[var_name]

        ioda_attr_data.close()
        ioda_hofx_data.close()

        nlocs = len(ioda_df.index)
        print('Number of locations in set: ' + str(nlocs))

        # Decode strings
        time = list(ioda_df['datetime'])

        for i in range(0,nlocs):
            temp = dt.datetime.strptime(time[i], '%Y-%m-%dT%H:%M:%SZ')
```

(continues on next page)

(continued from previous page)

```
            time[i] = temp.strftime('%Y%m%d_%H%M%S')

        ioda_df['datetime'] = time

        #set up MPR data
        mpr_data = []

        for var_name in hofx_vars:

            # Set up the needed columns
            ioda_df_var = ioda_df[['datetime','station_id',var_name+'@ObsType',
                                   'latitude','longitude','air_pressure',
                                   var_name+'@hofx',var_name+'@ObsValue',
                                   var_name+'@EffectiveQC']]

            # Cute down to locations with valid ObsValues
            ioda_df_var = ioda_df_var[abs(ioda_df_var[var_name+'@ObsValue']) < 1e6]
            nlocs = len(ioda_df_var.index)
            print(var_name+' has '+str(nlocs)+' valid obs.')

            # Add additional columns
            ioda_df_var['lead'] = '000000'
            ioda_df_var['MPR'] = 'MPR'
            ioda_df_var['nobs'] = nlocs
            ioda_df_var['index'] = range(0,nlocs)
            ioda_df_var['varname'] = var_name
            ioda_df_var['na'] = 'NA'

            # Arrange columns in MPR format
            cols = ['na','na','lead','datetime','datetime','lead','datetime',
                    'datetime','varname','na','lead','varname','na','na',
                    var_name+'@ObsType','na','na','lead','na','na','na','na','MPR',
                    'nobs','index','station_id','latitude','longitude',
                    'air_pressure','na',var_name+'@hofx',var_name+'@ObsValue',
                    var_name+'@EffectiveQC','na','na']

            ioda_df_var = ioda_df_var[cols]

            # Into a list and all to strings
            mpr_data = mpr_data + [list( map(str,i) ) for i in ioda_df_var.values.tolist() ]

            print("Total Length:\t" + repr(len(mpr_data)))

    except NameError:
        print("Can't find the input file.")
```

(continues on next page)

(continued from previous page)

```
        print("HofX variables in this file:\t" + repr(hofx_vars))
else:
    print("ERROR: read_iodav2_mpr.py -> Must specify input file.\n")
    sys.exit(1)

########################################################################
```

Running METplus

It is recommended to run this use case by:

Passing in StatAnalysis_fcstGFS_HofX_obsIODAv2_PyEmbed.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/StatAnalysis_fcstGFS_HofX_obsIODAv2_PyEmbed.conf -c /path/to/user_
↪system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in StatAnalysis_IODAv2 (relative to **OUTPUT_BASE**) and will contain the following file:

- dump.out

Keywords

Note:

- StatAnalysisToolUseCase
- PythonEmbeddingFileUseCase
- DataAssimilationUseCase
- IODA2NCToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/data_assimilation-StatAnalysis_fcstGFS_HofX_obsIODAv2_PyEmbed.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.5 Land Surface

Land Model diagnostics and verification against observations

7.2.16.5.1 PointStat: CESM and FLUXNET2015 Terrestrial Coupling Index (TCI)

model_applications/land_surface/PointStat_fcstCESM_obsFLUXNET2015_TCI.conf

Scientific Objective

This use case ingests two CESM (CAM and CLM) files and a new FLUX2015 dataset (NETCDF) that Paul Dirmeyer, GMU prepared from FLUXNET2015 ASCII dataset. The use case will calculate Terrestrial Coupling Index (TCI) from CESM datasets. Utilizing Python embedding, this use case taps into a new vital observation dataset and compares it to CESM simulations TCI forecast. Finally, it will generate plots of model TCI and observed TCI.

Datasets

Forecast: CESM 1979-1983 Simulations a. Community Land Model (CLM) and b. Community Atmosphere Model (CAM)

Observations: FLUXNET2015 post processed and converted to NETCDF. This data includes data collected from multiple regional flux towers.

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1057) section for more information.

Data Source: CESM - CGD; FLUXNET2015 - Paul Dirmeyer

METplus Components

This use case utilizes the METplus PyEmbedIngest to read the CESM files and calculate TCI using python embedding and a NETCDF file of the TCI is generated. The METplus PointStat processes the output of PyEmbedIngest and FLUXNET2015 dataset (using python embedding), and outputs the requested line types. Then METplus PlotPointObs tool reads the output of PyEmbedIngest and FLUXNET2015 dataset and produce plots of TCI from CESM and point observations. A custom loop runs through all the pre-defined seasons (DJF, MAM, JJA, SON) and runs PyEmbedIngest, PointStat, and PlotPointObs.

METplus Workflow

The PyEmbedIngest tool reads 2 CESM files containing Soil Moisture and Sensible Heat Flux, each composed of daily forecasts from 1979 to 1983 and calculates TCI and generates a NETCDF file of the TCI. One FLUXNET2015 NETCDF file containing station observations of several variables including Coupling Index of Soil Moisture and Sensible Heat Flux is read by Python Embedding.

Valid Beg: 1979-01-01 at 00z
Valid End: 1979-01-01 at 00z

PointStat is used to compare the two new fields (TCI calculated from CESM dataset and FLUXNET2015). Finally, PlotPointObs is run to plot the CESM TCI overlaying the FLUXNET2015 point observations.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. -c parm/use_cases/model_applications/land_surface/PointStat_fcstCESM_obsFLUXNET2015_TCI.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/land_surface/
↪PointStat_fcstCESM_obsFLUXNET2015_TCI.html
```

(continues on next page)

(continued from previous page)

```
# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

#############################################################
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
#############################################################

PROCESS_LIST = PyEmbedIngest,PointStat,PlotPointObs

#############################################################
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
# INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
# VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
#############################################################

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 1979060100
VALID_END = 1979060100
VALID_INCREMENT = 24H

LEAD_SEQ = 0

#############################################################
# Pre-determined seasons for the use case
#############################################################
CUSTOM_LOOP_LIST = DJF,MAM,JJA,SON

LOG_LEVEL=DEBUG

#############################################################
# PyEmbedIngest Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pyembedingest
#############################################################

PY_EMBED_INGEST_1_OUTPUT_DIR =
PY_EMBED_INGEST_1_OUTPUT_TEMPLATE = {OUTPUT_BASE}/regrid_data_plane_{custom}.nc
```

(continues on next page)

(continued from previous page)

```
#############################################################
# The last argument is the Latent Heat Flux.
# User can change it to use any other variable present in CESM files.
#############################################################

PY_EMBED_INGEST_1_SCRIPT = {PARM_BASE}/use_cases/model_applications/land_surface/PointStat_
↪fcstCESM_obsFLUXNET2015_TCI/cesm_tci.py {INPUT_BASE}/model_applications/land_surface/
↪PointStat_fcstCESM_obsFLUXNET2015_TCI/f.e21.FHIST.f09_f09_mg17.CESM2-CLM45physics.002.clm2.
↪h1.1979-83_SoilWater10cm.nc {INPUT_BASE}/model_applications/land_surface/PointStat_
↪fcstCESM_obsFLUXNET2015_TCI/f.e21.FHIST.f09_f09_mg17.CESM2-CLM45physics.002.cam.h1.1979-83_
↪CIvars.nc {custom} LHFLX

PY_EMBED_INGEST_1_TYPE = NUMPY
#PY_EMBED_INGEST_1_OUTPUT_GRID = G129
PY_EMBED_INGEST_1_OUTPUT_GRID = "latlon 288 192 -90.0 -0.0 0.9375 1.25"

#############################################################
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
#############################################################

FCST_POINT_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

#############################################################
# The default variables for the FLUXNET2015 observations id 10CM Soil Moisture
# and Latent Heat Flux. User can change them in the python Embedding script
# fluxnet_tci.py in the PointStat_fcstCESM_obsFLUXNET2015_TCI
# directory.
#############################################################
OBS_POINT_STAT_INPUT_TEMPLATE = PYTHON_NUMPY={PARM_BASE}/use_cases/model_applications/land_
↪surface/PointStat_fcstCESM_obsFLUXNET2015_TCI/fluxnet2015_tci.py {INPUT_BASE}/model_
↪applications/land_surface/PointStat_fcstCESM_obsFLUXNET2015_TCI/F2015_LoCo_AllChains_F2015.
↪nc4 {custom}

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/PointStat

#############################################################
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
#############################################################

POINT_STAT_ONCE_PER_FIELD = False

FCST_POINT_STAT_INPUT_DIR = {OUTPUT_BASE}/
```

(continues on next page)

(continued from previous page)

```
FCST_POINT_STAT_INPUT_TEMPLATE = regrid_data_plane_{custom}.nc
FCST_POINT_STAT_VAR1_LEVELS =
OBS_POINT_STAT_VAR1_NAME = TCI
OBS_POINT_STAT_VAR1_LEVELS = L0

FCST_POINT_STAT_VAR1_NAME = TCI_10cm_soil_depth
FCST_POINT_STAT_VAR1_LEVELS = Z10
BOTH_POINT_STAT_VAR1_THRESH =

#################################################################
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
#################################################################

LOG_POINT_STAT_VERBOSITY = 2

POINT_STAT_CONFIG_FILE ={PARM_BASE}/met_config/PointStatConfig_wrapped

POINT_STAT_INTERP_TYPE_METHOD = NEAREST
POINT_STAT_INTERP_TYPE_WIDTH = 1

POINT_STAT_OUTPUT_FLAG_CTC = BOTH
POINT_STAT_OUTPUT_FLAG_MPR = BOTH
POINT_STAT_OUTPUT_FLAG_CNT = BOTH

OBS_POINT_STAT_WINDOW_BEGIN = -1000000000
OBS_POINT_STAT_WINDOW_END = 2000000000

POINT_STAT_OFFSETS = 0

MODEL = CESM

POINT_STAT_DESC = TCI
OBTYPE =

POINT_STAT_REGRID_TO_GRID = NONE
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2

POINT_STAT_OBS_VALID_BEG = 19790101_000000
POINT_STAT_OBS_VALID_END = 20130101_000000

POINT_STAT_MASK_GRID = FULL
POINT_STAT_MASK_POLY =
POINT_STAT_MASK_SID =
```

(continues on next page)

(continued from previous page)

```
POINT_STAT_MESSAGE_TYPE = ADPSFC

POINT_STAT_OUTPUT_PREFIX = {custom}

################################################################
# PlotPointObs Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#plotpointobs
################################################################

PLOT_POINT_OBS_INPUT_TEMPLATE =PYTHON_NUMPY={PARM_BASE}/use_cases/model_applications/land_
↪surface/PointStat_fcstCESM_obsFLUXNET2015_TCI/fluxnet2015_tci.py {INPUT_BASE}/model_
↪applications/land_surface/PointStat_fcstCESM_obsFLUXNET2015_TCI/F2015_LoCo_AllChains_F2015.
↪nc4 {custom}

PLOT_POINT_OBS_GRID_INPUT_DIR = {OUTPUT_BASE}/
PLOT_POINT_OBS_GRID_INPUT_TEMPLATE = regrid_data_plane_{custom}.nc

PLOT_POINT_OBS_OUTPUT_DIR = {OUTPUT_BASE}/plot_point_obs
PLOT_POINT_OBS_OUTPUT_TEMPLATE = cesm_fluxnet2015_{custom}.ps

PLOT_POINT_OBS_TITLE = {custom} CESM vs FLUXNET2015 TCI

LOG_PLOT_POINT_OBS_VERBOSITY = 2

PLOT_POINT_OBS_GRID_DATA_FIELD = { name = "TCI_10cm_soil_depth"; level = "10cm_soil_depth"; }
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_PLOT_MIN = -40.0
PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_PLOT_MAX = 55.0

PLOT_POINT_OBS_POINT_DATA =
  {
    msg_typ = "ADPSFC";
    obs_thresh = >-9999;
    obs_var = "TCI";
    dotsize(x) = 3.5;
    line_color = [0, 0, 0];
    fill_plot_info = {
      flag = TRUE;
      color_table   = "MET_BASE/colortables/met_default.ctable";
      plot_min = -40.0;
      plot_max = 55.0;
      colorbar_flag = FALSE;
    }
  }
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
```

(continues on next page)

(continued from previous page)

```
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
```

(continues on next page)

(continued from previous page)

```
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses a Python embedding script to read input data

parm/use_cases/model_applications/land_surface/PointStat_fcstCESM_obsFLUXNET2015_TCI/cesm_tci.py

```
"""
Code adapted from Meg Fowler,CGD,NCAR
The code computes the Terrestrial Coupling Index (TCI)
from latent Heat Flux and 10 cm Soil Moisture
Designed to read Latent Heat Flux (from CAM) and Soil Temp (from CLM)
from two CESM files
User needs to provide the season (DJF, MAM, JJA, or SON) in the METplus conf file
User can change the variables to compute TCI
"""


import numpy as np
import xarray as xr
import pandas as pd
import datetime
import time
import sys
import os
import netCDF4 as nc

if len(sys.argv) < 4:
    print("Must specify the following elements: sfc_flux_file soil_file season (DJF, MAM,␣
↪JJA, or SON)")
    sys.exit(1)

fileCLM = os.path.expandvars(sys.argv[1])
fileCAM = os.path.expandvars(sys.argv[2])
season = sys.argv[3]
var_y = sys.argv[4]

print('Starting Terrestrial Coupling Index Calculation for: ',season)

camDS_CLM45         = xr.open_dataset(fileCAM, decode_times=False)
print('Finished reading in CAM file')
clmDS_CLM45         = xr.open_dataset(fileCLM, decode_times=False)
print('Finished reading in CLM file')
```

(continues on next page)

(continued from previous page)

```
if season=="DJF":
    ss = 0
elif season=="MAM":
    ss = 1
elif season=="JJA":
    ss = 2
elif season=="SON":
    ss = 3
else:
    sys.exit('ERROR  : URECOGNIZED SEASON, PLEASE USE DJF, MAM, JJA, OR SON')

units, reference_date = camDS_CLM45.time.attrs['units'].split('since')
camDS_CLM45['time'] = pd.date_range(start=reference_date, periods=camDS_CLM45.sizes['time'],␣
↪freq='D')
camDS_time=camDS_CLM45.time[0]
dt_object = datetime.datetime.utcfromtimestamp(camDS_time.values.tolist()/1e9)
vDate=dt_object.strftime("%Y%m%d")


units, reference_date = clmDS_CLM45.time.attrs['units'].split('since')
clmDS_CLM45['time'] = pd.date_range(start=reference_date, periods=clmDS_CLM45.sizes['time'],␣
↪freq='D')

ds = camDS_CLM45
ds['SOILWATER_10CM'] = (('time','lat','lon'), clmDS_CLM45.SOILWATER_10CM.values)

xname = 'SOILWATER_10CM'    # Controlling variable
#yname = 'LHFLX'             # Responding variable
yname=str(var_y)

xday = ds[xname].groupby('time.season')
yday = ds[yname].groupby('time.season')

# Get the covariance of the two (numerator in coupling index)
covarTerm = ((xday - xday.mean()) * (yday - yday.mean())).groupby('time.season').sum() /␣
↪xday.count()

# Now compute the coupling index
couplingIndex = covarTerm/xday.std()
ci_season=couplingIndex[ss,:,:]

ci = np.where(np.isnan(ci_season), -9999, ci_season)

met_data = ci[:,:]
```

(continues on next page)

(continued from previous page)

```
met_data = met_data[::-1].copy()

#trim the lat/lon grids so they match the data fields
lat_met = camDS_CLM45.lat
lon_met = camDS_CLM45.lon
print(" Model Data shape: "+repr(met_data.shape))
v_str = vDate
v_str = v_str + '_000000'
#print(" Valid date: "+v_str)
lat_ll = float(lat_met.min())
lon_ll = float(lon_met.min())
n_lat = lat_met.shape[0]
n_lon = lon_met.shape[0]
delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)

print(f"variables:"
        f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat: {delta_
↪lat} delta_lon: {delta_lon}")

attrs = {
        'valid': v_str,
        'init': v_str,
        'lead': "000000",
        'accum': "000000",
        'name': 'TCI',
        'standard_name': 'terrestrial_coupling_index',
        'long_name': 'terrestrial_coupling_index',
        'level': "10cm_soil_depth",
        'units': "W/m2",

        'grid': {
            'type': "LatLon",
            'name': "CESM Grid",
            'lat_ll': lat_ll,
            'lon_ll': lon_ll,
            'delta_lat': delta_lat,
            'delta_lon': delta_lon,
            'Nlat': n_lat,
            'Nlon': n_lon,
            }
        }
```

parm/use_cases/model_applications/land_surface/PointStat_fcstCESM_obsFLUXNET2015_TCI/fluxnet2015_tci.py

```
"""
This python embedding code reads in the FLUXNET2015 NETCDF data
and passed to point_data. User need to pass on the season.
User can change the verifying variable
"""


import numpy
import sys
import os
import glob
import numpy as np
import xarray as xr
import datetime
from datetime import date, timedelta
import pandas as pd


if len(sys.argv) < 2:
    print("Must specify the following elements: FLUXNET2015_file season")
    sys.exit(1)

obsfile = os.path.expandvars(sys.argv[1])
season = sys.argv[2]

f2015data         = xr.open_dataset(obsfile, decode_times=False)

if season=="DJF":
    ss = 0
elif season=="MAM":
    ss = 1
elif season=="JJA":
    ss = 2
elif season=="SON":
    ss = 3
else:
    print('Unrecognized season, please use DJF, MAM, JJA, SON')
    exit()

start_year=f2015data['Start year'].values.tolist()
end_year=f2015data['End year'].values.tolist()
vld=pd.to_datetime(start_year,format='%Y')
vld=vld.strftime("%Y%m%d")
vld=vld+ '_000000'

sid=f2015data['Station'].values.tolist()
```

(continues on next page)

(continued from previous page)

```
print("Length :",len(sid))
print("SID :",sid)

lat=f2015data['Latitude'].values.tolist()
lon=f2015data['Longitude'].values.tolist()
# User can change the name of the variable below
obs=f2015data['CI Sfc_SM Latent_Heat'].values.tolist()
obs=np.array(obs)
obs=obs[:,ss]


#create dummy lists for the message type, elevation, variable name, level, height, and qc␣
↪string
#numpy is more efficient at creating the lists, but need to convert to Pythonic lists
typ = np.full(len(sid),'ADPSFC').tolist()
elv = np.full(len(sid),10).tolist()
var = np.full(len(sid),'TCI').tolist()
lvl = np.full(len(sid),10).tolist()
hgt = np.zeros(len(sid),dtype=int).tolist()
qc = np.full(len(sid),'NA').tolist()
obs = np.where(np.isnan(obs), -9999, obs)
obs = np.full(len(sid),obs).tolist()
vld = np.full(len(sid),vld).tolist()

l_tuple = list(zip(typ,sid,vld,lat,lon,elv,var,lvl,hgt,qc,obs))
point_data = [list(ele) for ele in l_tuple]

#print("Data Length:\t" + repr(len(point_data)))
#print("Data Type:\t" + repr(type(point_data)))

#print(" Point Data Shape: ",np.shape(point_data))
print(" Point Data: ",point_data)
```

Running METplus

Pass the use case configuration file to the run_metplus.py script along with any user-specific system configuration files if desired:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/land_surface/PointStat_
↪fcstCESM_obsFLUXNET2015_TCI.conf /path/to/user_system.conf
```

See *Running METplus* (page 26) for more information.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for the use case will be found in 3 folders(relative to **OUTPUT_BASE**). Those folders are:

- PyEmbedIngest

The **OUTPUT_BASE** folder contains all of the TCI output calculated using CESM files in NETCDF format:

- regrid_data_plane_DJF.nc
- regrid_data_plane_JJA.nc
- regrid_data_plane_MAM.nc
- regrid_data_plane_SON.nc
- PointStat

The final folder, PointStat, contains all of the following output from the PointStat call:

- point_stat_DJF_000000L_19790101_000000V_cnt.txt
- point_stat_DJF_000000L_19790101_000000V_ctc.txt
- point_stat_DJF_000000L_19790101_000000V_mpr.txt
- point_stat_DJF_000000L_19790101_000000V.stat
- point_stat_JJA_000000L_19790101_000000V_cnt.txt
- point_stat_JJA_000000L_19790101_000000V_ctc.txt
- point_stat_JJA_000000L_19790101_000000V_mpr.txt
- point_stat_JJA_000000L_19790101_000000V.stat
- point_stat_MAM_000000L_19790101_000000V_cnt.txt
- point_stat_MAM_000000L_19790101_000000V_ctc.txt
- point_stat_MAM_000000L_19790101_000000V_mpr.txt
- point_stat_MAM_000000L_19790101_000000V.stat
- point_stat_SON_000000L_19790101_000000V_cnt.txt
- point_stat_SON_000000L_19790101_000000V_ctc.txt
- point_stat_SON_000000L_19790101_000000V_mpr.txt
- point_stat_SON_000000L_19790101_000000V.stat
- PlotPointObs

The final folder plot_point_obs, contains all of the plots from the PlotPointObs call:

- cesm_fluxnet2015_DJF.ps
- cesm_fluxnet2015_JJA.ps
- cesm_fluxnet2015_MAM.ps
- cesm_fluxnet2015_SON.ps

Keywords

Note:

- PyEmbedIngestToolUseCase
- PointStatToolUseCase
- PlotPointObsToolUseCase
- PythonEmbeddingFileUseCase
- NETCDFFileUseCase
- LandSurfaceAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/land_surface-PointStat_fcstCESM_obsFLUXNET2015_TCI.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.6 Marine and Cryosphere

Data related to verification involving marine and cryosphere systems, which includes sea-ice

7.2.16.6.1 GridStat: Python Embedding for sea surface salinity using level 3, 1 day composite obs

model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss.conf

Scientific Objective

This use case utilizes Python embedding to extract several statistics from the sea surface salinity data over the globe, which was already being done in a closed system. By producing the same output via METplus, this use case provides standardization and reproducible results.

Datasets

Forecast: RTOFS sss file via Python Embedding script/file

Observations: SMOS sss file via Python Embedding script/file

Sea Ice Masking: RTOFS ice cover file via Python Embedding script/file

Climatology: WOA sss file via Python Embedding script/file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1076) section for more information.

Data Source: JPL's PODAAC and NCEP's FTPPRD data servers

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- scikit-learn
- pyresample

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

```
[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python
```

METplus Components

This use case utilizes the METplus GridStat wrapper to generate a command to run the MET tool GridStat with Python Embedding for the specified user hemispheres

METplus Workflow

GridStat is the only tool called in this example. This use case will pass in both the observation, forecast, and climatology gridded data being pulled from the files via Python Embedding. All of the desired statistics reside in the CNT line type, so that is the only output requested. It processes the following run time:

Valid: 2021-05-03 0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
```

(continues on next page)

(continued from previous page)

```
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG=20210503
VALID_END=20210503
VALID_INCREMENT = 1M

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

GRID_STAT_ONCE_PER_FIELD = False

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsSMOS_climWOA_sss

FCST_VAR1_NAME = {CONFIG_DIR}/read_rtofs_smos_woa.py {INPUT_BASE}/model_applications/marine_
↪and_cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss/{valid?fmt=%Y%m%d}_rtofs_glo_2ds_
↪f024_prog.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_
↪obsSMOS_climWOA_sss/SMOS-L3-GLOB_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/
```

(continues on next page)

(continued from previous page)

```
↪marine_and_cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss/OSTIA-UKMO-L4-GLOB-v2.0_
↪{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsSMOS_climWOA_sss {valid?fmt=%Y%m%d} fcst

OBS_VAR1_NAME = {CONFIG_DIR}/read_rtofs_smos_woa.py {INPUT_BASE}/model_applications/marine_
↪and_cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss/{valid?fmt=%Y%m%d}_rtofs_glo_2ds_
↪f024_prog.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_
↪obsSMOS_climWOA_sss/SMOS-L3-GLOB_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/
↪marine_and_cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss/OSTIA-UKMO-L4-GLOB-v2.0_
↪{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsSMOS_climWOA_sss {valid?fmt=%Y%m%d} obs

GRID_STAT_CLIMO_MEAN_FIELD = {name="{CONFIG_DIR}/read_rtofs_smos_woa.py {INPUT_BASE}/model_
↪applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss/{valid?fmt=%Y%m
↪%d}_rtofs_glo_2ds_f024_prog.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪GridStat_fcstRTOFS_obsSMOS_climWOA_sss/SMOS-L3-GLOB_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/
↪model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss/OSTIA-UKMO-
↪L4-GLOB-v2.0_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪GridStat_fcstRTOFS_obsSMOS_climWOA_sss {valid?fmt=%Y%m%d} climo"; level="(*,*)";}


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_REGRID_TO_GRID = NONE

MODEL = RTOFS
OBTYPE = SMOS

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_OUTPUT_PREFIX = SSS

GRID_STAT_OUTPUT_FLAG_CNT = BOTH
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
```

(continues on next page)

(continued from previous page)

```
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
```

(continues on next page)

(continued from previous page)

```
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses one Python script to read forecast and observation data

parm/use_cases/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss/read_rtof

```
#!/bin/env python
"""
Code adapted from
Todd Spindler
NOAA/NWS/NCEP/EMC
Designed to read in RTOFS,SMOS,WOA and OSTIA data
and based on user input, read sss data
and pass back in memory the forecast, observation, or climatology
data field
"""
```

(continues on next page)

(continued from previous page)

```
import numpy as np
import xarray as xr
import pandas as pd
import pyresample as pyr
from pandas.tseries.offsets import DateOffset
from datetime import datetime, timedelta
from sklearn.metrics import mean_squared_error
import io
from glob import glob
import warnings
import os, sys


if len(sys.argv) < 6:
    print("Must specify the following elements: fcst_file obs_file ice_file, climo_file,␣
↪valid_date, file_flag")
    sys.exit(1)

rtofsfile = os.path.expandvars(sys.argv[1])
sssfile = os.path.expandvars(sys.argv[2])
icefile = os.path.expandvars(sys.argv[3])
climoDir = os.path.expandvars(sys.argv[4])
vDate=datetime.strptime(sys.argv[5],'%Y%m%d')
file_flag = sys.argv[6]

print('Starting Satellite SMOS V&V at',datetime.now(),'for',vDate, ' file_flag:',file_flag)

pd.date_range(vDate,vDate)
platform='SMOS'
param='sss'


######################################################################
# READ SMOS data ###################################################
######################################################################

if not os.path.exists(sssfile):
        print('missing SMOS file for',vDate)

sss_data=xr.open_dataset(sssfile,decode_times=True)
sss_data['time']=sss_data.time-pd.Timedelta('12H')  # shift 12Z offset time to 00Z
sss_data2=sss_data['sss'].astype('single')
print('Retrieved SMOS data from NESDIS for',sss_data2.time.values)
sss_data2=sss_data2.rename({'longitude':'lon','latitude':'lat'})
```

(continues on next page)

(continued from previous page)

```
# all coords need to be single precision
sss_data2['lon']=sss_data2.lon.astype('single')
sss_data2['lat']=sss_data2.lat.astype('single')
sss_data2.attrs['platform']=platform
sss_data2.attrs['units']='PSU'

######################################################################
# READ RTOFS data (model output in Tri-polar coordinates) ############
######################################################################

print('reading rtofs ice')
if not os.path.exists(rtofsfile):
    print('missing rtofs file',rtofsfile)
    sys.exit(1)

indata=xr.open_dataset(rtofsfile,decode_times=True)


indata=indata.mean(dim='MT')
indata = indata[param][:-1,]
indata.coords['time']=vDate
#indata.coords['fcst']=fcst

outdata=indata.copy()

outdata=outdata.rename({'Longitude':'lon','Latitude':'lat',})
# all coords need to be single precision
outdata['lon']=outdata.lon.astype('single')
outdata['lat']=outdata.lat.astype('single')
outdata.attrs['platform']='rtofs '+platform

######################################################################
# READ CLIMO WOA data - May require 2 files depending on the date ###
######################################################################

if not os.path.exists(climoDir):
        print('missing climo file file for',vDate)

vDate=pd.Timestamp(vDate)

climofile="woa13_decav_s{:02n}_04v2.nc".format(vDate.month)
climo_data=xr.open_dataset(climoDir+'/'+climofile,decode_times=False)
climo_data=climo_data['s_an'].squeeze()[0,]
```

(continues on next page)

(continued from previous page)

```
if vDate.day==15:  # even for Feb, just because
    climofile="woa13_decav_s{:02n}_04v2.nc".format(vDate.month)
    climo_data=xr.open_dataset(climoDir+'/'+climofile,decode_times=False)
    climo_data=climo_data['s_an'].squeeze()[0,]  # surface only
else:
    if vDate.day < 15:
        start=vDate - DateOffset(months=1,day=15)
        stop=pd.Timestamp(vDate.year,vDate.month,15)
    else:
        start=pd.Timestamp(vDate.year,vDate.month,15)
        stop=vDate + DateOffset(months=1,day=15)
    left=(vDate-start)/(stop-start)

    climofile1="woa13_decav_s{:02n}_04v2.nc".format(start.month)
    climofile2="woa13_decav_s{:02n}_04v2.nc".format(stop.month)
    climo_data1=xr.open_dataset(climoDir+'/'+climofile1,decode_times=False)
    climo_data2=xr.open_dataset(climoDir+'/'+climofile2,decode_times=False)
    climo_data1=climo_data1['s_an'].squeeze()[0,]  # surface only
    climo_data2=climo_data2['s_an'].squeeze()[0,]  # surface only

    print('climofile1 :', climofile1)
    print('climofile2 :', climofile2)
    climo_data=climo_data1+((climo_data2-climo_data1)*left)
    climofile='weighted average of '+climofile1+' and '+climofile2

# all coords need to be single precision
climo_data['lon']=climo_data.lon.astype('single')
climo_data['lat']=climo_data.lat.astype('single')
climo_data.attrs['platform']='woa'
climo_data.attrs['filename']=climofile

######################################################################
# READ ICE data for masking ##########################################
######################################################################

if not os.path.exists(icefile):
        print('missing OSTIA ice file for',vDate)

ice_data=xr.open_dataset(icefile,decode_times=True)
ice_data=ice_data.rename({'sea_ice_fraction':'ice'})

# all coords need to be single precision
ice_data2=ice_data.ice.astype('single')
ice_data2['lon']=ice_data2.lon.astype('single')
```

(continues on next page)

(continued from previous page)

```
ice_data2['lat']=ice_data2.lat.astype('single')


def regrid(model,obs):
    """
    regrid data to obs -- this assumes DataArrays
    """
    model2=model.copy()
    model2_lon=model2.lon.values
    model2_lat=model2.lat.values
    model2_data=model2.to_masked_array()
    if model2_lon.ndim==1:
        model2_lon,model2_lat=np.meshgrid(model2_lon,model2_lat)

    obs2=obs.copy()
    obs2_lon=obs2.lon.astype('single').values
    obs2_lat=obs2.lat.astype('single').values
    obs2_data=obs2.astype('single').to_masked_array()
    if obs2.lon.ndim==1:
        obs2_lon,obs2_lat=np.meshgrid(obs2.lon.values,obs2.lat.values)

    model2_lon1=pyr.utils.wrap_longitudes(model2_lon)
    model2_lat1=model2_lat.copy()
    obs2_lon1=pyr.utils.wrap_longitudes(obs2_lon)
    obs2_lat1=obs2_lat.copy()

    # pyresample gausssian-weighted kd-tree interp
    # define the grids
    orig_def = pyr.geometry.GridDefinition(lons=model2_lon1,lats=model2_lat1)
    targ_def = pyr.geometry.GridDefinition(lons=obs2_lon1,lats=obs2_lat1)
    radius=50000
    sigmas=25000
    model2_data2=pyr.kd_tree.resample_gauss(orig_def,model2_data,targ_def,
                                            radius_of_influence=radius,
                                            sigmas=sigmas,
                                            fill_value=None)
    model=xr.DataArray(model2_data2,coords=[obs.lat.values,obs.lon.values],dims=['lat','lon
↪'])

    return model

def expand_grid(data):
    """
    concatenate global data for edge wraps
    """
```

(continues on next page)

(continued from previous page)

```
    data2=data.copy()
    data2['lon']=data2.lon+360
    data3=xr.concat((data,data2),dim='lon')
    return data3

sss_data2=sss_data2.squeeze()

print('regridding climo to obs')
climo_data=climo_data.squeeze()
climo_data=regrid(climo_data,sss_data2)

print('regridding ice to obs')
ice_data2=regrid(ice_data2,sss_data2)

print('regridding model to obs')
model2=regrid(outdata,sss_data2)

# combine obs ice mask with ncep
obs2=sss_data2.to_masked_array()
ice2=ice_data2.to_masked_array()
climo2=climo_data.to_masked_array()
model2=model2.to_masked_array()

#reconcile with obs
obs2.mask=np.ma.mask_or(obs2.mask,ice2>0.0)
obs2.mask=np.ma.mask_or(obs2.mask,climo2.mask)
obs2.mask=np.ma.mask_or(obs2.mask,model2.mask)
climo2.mask=obs2.mask
model2.mask=obs2.mask

obs2=xr.DataArray(obs2,coords=[sss_data2.lat.values,sss_data2.lon.values], dims=['lat','lon
↪'])
model2=xr.DataArray(model2,coords=[sss_data2.lat.values,sss_data2.lon.values], dims=['lat',
↪'lon'])
climo2=xr.DataArray(climo2,coords=[sss_data2.lat.values,sss_data2.lon.values], dims=['lat',
↪'lon'])

model2=expand_grid(model2)
climo2=expand_grid(climo2)
obs2=expand_grid(obs2)

#Create the MET grids based on the file_flag
if file_flag == 'fcst':
    met_data = model2[:,:]
```

(continues on next page)

(continued from previous page)

```
    #trim the lat/lon grids so they match the data fields
    lat_met = model2.lat
    lon_met = model2.lon
    print(" RTOFS Data shape: "+repr(met_data.shape))
    v_str = vDate.strftime("%Y%m%d")
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'sss',
            'standard_name': 'sss',
            'long_name': 'sss',
            'level': "SURFACE",
            'units': "degC",

            'grid': {
                'type': "LatLon",
                'name': "RTOFS Grid",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs

if file_flag == 'obs':
    met_data = obs2[:,:]
    #trim the lat/lon grids so they match the data fields
    lat_met = obs2.lat
    lon_met = obs2.lon
    v_str = vDate.strftime("%Y%m%d")
```

(continues on next page)

(continued from previous page)

```
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'sss',
            'standard_name': 'analyzed sss',
            'long_name': 'analyzed sss',
            'level': "SURFACE",
            'units': "degC",

            'grid': {
                'type': "LatLon",
                'name': "Lat Lon",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs

if file_flag == 'climo':
    met_data = climo2[:,:]
    #modify the lat and lon grids since they need to match the data dimensions, and code␣
↪cuts the last row/column of data
    lat_met = climo2.lat
    lon_met = climo2.lon
    v_str = vDate.strftime("%Y%m%d")
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
```

(continues on next page)

(continued from previous page)

```
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'sea_water_temperature',
            'standard_name': 'sea_water_temperature',
            'long_name': 'sea_water_temperature',
            'level': "SURFACE",
            'units': "degC",

            'grid': {
                'type': "LatLon",
                'name': "crs Grid",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstRTOFS_obsSMOS_climWOA_sss.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstRTOFS_obsSMOS_climWOA_sss.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsSMOS_climWOA_sss.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for thisIce use case will be found in 20210503 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_SSS_000000L_20210503_000000V.stat
- grid_stat_SSS_000000L_20210503_000000V_cnt.txt
- grid_stat_SSS_000000L_20210503_000000V_pairs.nc

Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase
- ClimatologyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/marine_and_cryosphere-GridStat_fcstRTOFS_obsSMOS_climWOA_sss.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.6.2 Grid-Stat and MODE: Sea Ice Validation

model_applications/marine_and_cryosphere/GridStat_MODE_fcstIMS _obsNCEP_sea_ice.conf

Scientific Objective

Run Grid-Stat and MODE to compare the National Ice Center (NIC) Interactive Multisensor Snow and Ice Mapping System (IMS) and the National Centers for Environmental Prediction (NCEP) sea ice analysis. This is a validation and diagnostics use case because it is limited to a comparison between IMS analysis to NCEP analysis. Written by Lindsay Blank, NCAR. January 2020

Datasets

Both IMS and NCEP sea ice analyses are observation datasets. For the purposes of MET, IMS is referred to as "forecast" and NCEP is referred to as "observation".

- **Forecast dataset: IMS Sea Ice Concentration**
  - Variable of interest: ICEC; ICEC is a binary field where "1" means a sea ice concentration of >=0.40 and "0" means a sea ice concentration of <0.40.
  - Level: Z0 (surface)
  - Dates: 20190201 - 20190228
  - Valid time: 22 UTC
  - Format: Grib2
  - Projection: 4-km Polar Stereographic
- **Observation dataset: NCEP Sea Ice Concentration**
  - Variable of interest: ICEC; ICEC is the sea ice concentration with values from 0.0 - 1.0. Values >1.0 && <=1.28 indicate flagged data to be included and should be set to ==1.0 when running MET. Values >1.28 should be ignored as that indicates an invalid observation.
  - Level: Z0 (surface)
  - Dates: 20190201 - 20190228
  - Valid time: 00 UTC
  - Format: Grib2
  - Projection: 12.7-km Polar Stereographic

- Data source: Received from Robert Grumbine at EMC. IMS data is originally from the NIC. NCEP data is originally from NCEP.
- Location: IMS: https://www.natice.noaa.gov/ims/index.html; IMS - (https://polar.ncep.noaa.gov/seaice/Analyses.shtml)

METplus Components

This use case runs the MET GridStat and MODE tools.

METplus Workflow

The workflow processes the data by valid time, meaning that each tool will be run for each time before moving onto the next valid time. The GridStat tool is called first followed by the MODE tool. It processes analysis times from 2019-02-01 to 2019-02-05. The valid times for each analysis are different from one another (please see *Datasets* (page 1078) section for more information).

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config. Then, it loads any configuration files passed to METplus by the command line with the -c option.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/marine_and_
↪cryosphere/GridStat_MODE_fcstIMS_obsNCEP_sea_ice.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat, Mode


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
```

(continues on next page)

(continued from previous page)

```
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###


LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG=20190201
VALID_END=20190201
VALID_INCREMENT=86400


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/IMS_
↪data
FCST_GRID_STAT_INPUT_TEMPLATE = imssnow96.{valid?fmt=%Y%m%d}.grb.grib2

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/NCEP_
↪data
OBS_GRID_STAT_INPUT_TEMPLATE = seaice.t00z.north12psg.grib2.{valid?fmt=%Y%m%d}

GRID_STAT_VERIFICATION_MASK_TEMPLATE = {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪sea_ice/seaice_nland127.nc

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/
↪GridStat
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/grid_stat


FCST_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/IMS_data
FCST_MODE_INPUT_TEMPLATE = imssnow96.{valid?fmt=%Y%m%d}.grb.grib2

OBS_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/NCEP_data
OBS_MODE_INPUT_TEMPLATE = seaice.t00z.north12psg.grib2.{valid?fmt=%Y%m%d}

MODE_VERIFICATION_MASK_TEMPLATE = {INPUT_BASE}/model_applications/marine_and_cryosphere/sea_
↪ice/seaice_nland127.nc
```

(continues on next page)

(continued from previous page)

```
MODE_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/MODE
MODE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/mode


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = IMS
OBTYPE = NCEP

FCST_VAR1_NAME = ICEC
FCST_VAR1_LEVELS = Z0
FCST_VAR1_THRESH = ==1.0

FCST_MODE_CONV_THRESH = ==1.00
OBS_MODE_CONV_THRESH = >=0.40


OBS_VAR1_NAME = ICEC
OBS_VAR1_LEVELS = Z0
OBS_VAR1_THRESH = >=0.40
OBS_VAR1_OPTIONS = censor_thresh = [ >1.00 && <=1.28, >1.28 ]; censor_val    = [ 1.00 , -
↪9999 ];

MODE_OBS_CENSOR_THRESH = >1.00 && <=1.28, >1.28
MODE_OBS_CENSOR_VAL = 1.00 , -9999


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9

GRID_STAT_REGRID_TO_GRID = OBS

GRID_STAT_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}_
↪{CURRENT_FCST_LEVEL}

GRID_STAT_MASK_GRID =

GRID_STAT_OUTPUT_FLAG_CTC = STAT
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_FHO = STAT
GRID_STAT_OUTPUT_FLAG_CNT = STAT
GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT
GRID_STAT_OUTPUT_FLAG_PCT = STAT
GRID_STAT_OUTPUT_FLAG_PSTD = STAT
GRID_STAT_OUTPUT_FLAG_NBRCNT = STAT

GRID_STAT_NC_PAIRS_FLAG_NBRHD = TRUE


###
# MODE Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mode
###

MODE_REGRID_TO_GRID = OBS

MODE_QUILT = False

MODE_CONV_RADIUS = 50

MODE_MERGE_FLAG = NONE
MODE_MERGE_THRESH = >=1.25

MODE_GRID_RES = 12.7

MODE_MASK_MISSING_FLAG = BOTH

MODE_MATCH_FLAG = NO_MERGE

MODE_MASK_POLY_FLAG = BOTH

MODE_TOTAL_INTEREST_THRESH = 0.8

MODE_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

GridStatConfig_wrapped

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
```

(continues on next page)

(continued from previous page)

```
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}
```

(continues on next page)

(continued from previous page)

```
//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
```

(continues on next page)

(continued from previous page)

```
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
```

(continues on next page)

(continued from previous page)

```
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

MODEConfig_wrapped

Note: See the *MODE MET Configuration* (page 174) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Approximate grid resolution (km)
//
// grid_res =
${METPLUS_GRID_RES}

////////////////////////////////////////////////////////////////////////////////

//
// Run all permutations of radius and threshold
//
// quilt =
${METPLUS_QUILT}

//
// MODE Multivar boolean combination logic
//
//multivar_logic =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_MULTIVAR_LOGIC}

//multivar_intensity_flag =
${METPLUS_MULTIVAR_INTENSITY_FLAG}

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FIELD}

   ${METPLUS_FCST_CENSOR_THRESH}
   ${METPLUS_FCST_CENSOR_VAL}
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}
   ${METPLUS_FCST_VLD_THRESH}
   ${METPLUS_FCST_FILTER_ATTR_NAME}
   ${METPLUS_FCST_FILTER_ATTR_THRESH}
   ${METPLUS_FCST_MERGE_THRESH}
   ${METPLUS_FCST_MERGE_FLAG}
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_MULTIVAR_NAME}
   ${METPLUS_FCST_MULTIVAR_LEVEL}
}

obs = {
   ${METPLUS_OBS_FIELD}

   ${METPLUS_OBS_CENSOR_THRESH}
   ${METPLUS_OBS_CENSOR_VAL}
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
   ${METPLUS_OBS_VLD_THRESH}
   ${METPLUS_OBS_FILTER_ATTR_NAME}
   ${METPLUS_OBS_FILTER_ATTR_THRESH}
   ${METPLUS_OBS_MERGE_THRESH}
   ${METPLUS_OBS_MERGE_FLAG}
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_MULTIVAR_NAME}
   ${METPLUS_OBS_MULTIVAR_LEVEL}
}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Handle missing data
//
// mask_missing_flag =
${METPLUS_MASK_MISSING_FLAG}

//
// Match objects between the forecast and observation fields
//
//match_flag =
${METPLUS_MATCH_FLAG}

//
// Maximum centroid distance for objects to be compared
//
//max_centroid_dist =
${METPLUS_MAX_CENTROID_DIST}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine weights
//
//weight = {
${METPLUS_WEIGHT_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine interest functions
//
interest_function = {

   ${METPLUS_INTEREST_FUNCTION_CENTROID_DIST}

   ${METPLUS_INTEREST_FUNCTION_BOUNDARY_DIST}

   ${METPLUS_INTEREST_FUNCTION_CONVEX_HULL_DIST}
```

(continues on next page)

(continued from previous page)

```
   angle_diff = (
      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 0.0 )
   );

   aspect_diff = (
      (  0.00, 1.0 )
      (  0.10, 1.0 )
      (  0.75, 0.0 )
   );

   corner   = 0.8;
   ratio_if = (
      (    0.0, 0.0 )
      ( corner, 1.0 )
      (    1.0, 1.0 )
   );

   area_ratio = ratio_if;

   int_area_ratio = (
      ( 0.00, 0.00 )
      ( 0.10, 0.50 )
      ( 0.25, 1.00 )
      ( 1.00, 1.00 )
   );

   curvature_ratio = ratio_if;

   complexity_ratio = ratio_if;

   inten_perc_ratio = ratio_if;
}

////////////////////////////////////////////////////////////////////////////////

//
// Total interest threshold for determining matches
//
//total_interest_thresh =
${METPLUS_TOTAL_INTEREST_THRESH}

//
```

(continues on next page)

(continued from previous page)

```
// Interest threshold for printing output pair information
//
print_interest_thresh = 0.0;

////////////////////////////////////////////////////////////////////////////////

//
// Plotting information
//
met_data_dir = "MET_BASE";

fcst_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

obs_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

object_plot = {
   color_table      = "MET_BASE/colortables/mode_obj.ctable";
}

//
// Boolean for plotting on the region of valid data within the domain
//
plot_valid_flag = FALSE;

//
// Plot polyline edges using great circle arcs instead of straight lines
//
plot_gcarc_flag = FALSE;

////////////////////////////////////////////////////////////////////////////////

//
// NetCDF matched pairs, PostScript, and contingency table output files
//
//ps_plot_flag =
${METPLUS_PS_PLOT_FLAG}
```

(continues on next page)

(continued from previous page)

```
//nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

//ct_stats_flag =
${METPLUS_CT_STATS_FLAG}


////////////////////////////////////////////////////////////////////////////////

shift_right = 0;   //  grid squares

////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_MODE_fcstIMS_obsNCEP_sea_ice.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/GridStat_MODE_fcstIMS_obsNCEP_sea_ice.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_MODE_fcstIMS_obsNCEP_sea_ice.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/GridStat_MODE_fcstIMS_obsNCEP_sea_ice.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - PAth to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE All of these items must be found under the [dir] section.

Expected Output

A successful run of this use case will output the following to the screen and logfile:

```
INFO: METplus has successfully finished running.
```

A successful run will have the following output files in the location defined by {OUTPUT_BASE}, which is located in the metplus_system.conf configuration file located in /path/to/METplus/parm/metplus_config. This list of files should be found for every time run through METplus. GridStat output will be in model_applications/marine_and_cryosphere/sea_ice/GridStat relative to the {OUTPUT_BASE}. MODE output will be in model_applications/marine_and_cryosphere/sea_ice/MODE relative to the {OUTPUT_BASE}. Using the output for 20190201 as an example:

GridStat output:

- grid_stat_IMS_ICEC_vs_NCEP_ICEC_Z0_000000L_20190201_220000V_pairs.nc
- grid_stat_IMS_ICEC_vs_NCEP_ICEC_Z0_000000L_20190201_220000V.stat

MODE output:

- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R1_T1_cts.txt
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R1_T1_obj.nc
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R1_T1_obj.txt
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R1_T1.ps
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R2_T1_cts.txt
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R2_T1_obj.nc
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R2_T1_obj.txt
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R2_T1.ps
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R3_T1_cts.txt
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R3_T1_obj.nc
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R3_T1_obj.txt
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R3_T1.ps

- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R4_T1_cts.txt
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R4_T1_obj.nc
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R4_T1_obj.txt
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R4_T1.ps
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R5_T1_cts.txt
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R5_T1_obj.nc
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R5_T1_obj.txt
- mode_IMS_ICEC_vs_NCEP_ICEC_000000L_20190201_220000V_000000A_R5_T1.ps

Keywords

Note:

- GridStatToolUseCase
- MODEToolUseCase
- MarineAndCryosphereAppUseCase
- ValidationUseCase
- S2SAppUseCase
- NOAAEMCOrgUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/marine_and_cryosphere_GridStat_MODE_fcstIMS_obsNCEP_Sea_Ice.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.6.3 GridStat: Python Embedding to read and process SST

model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst.conf

Scientific Objective

This use case utilizes Python embedding to extract several statistics from the sea surface temperature data over the globe, which was already being done in a closed system. By producing the same output via METplus, this use case provides standardization and reproducible results.

Datasets

Forecast: RTOFS sst file via Python Embedding script/file

Observations: GHRSST sst file via Python Embedding script/file

Sea Ice Masking: RTOFS ice cover file via Python Embedding script/file

Climatology: WOA sst file via Python Embedding script/file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1113) section for more information.

Data Source: JPL's PODAAC and NCEP's FTPPRD data servers

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- scikit-learn
- pyresample

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case utilizes the METplus GridStat wrapper to generate a command to run the MET tool GridStat with Python Embedding each time a field (fcst, obs, and climo) is needed.

METplus Workflow

GridStat is the only tool called in this example. This use case will pass in both the observation, forecast, and climatology gridded data being pulled from the files via Python Embedding. All of the desired statistics reside in the CNT line type, so that is the only output requested. It processes the following run time:

Valid: 2021-05-03 0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG=20210503
VALID_END=20210503
VALID_INCREMENT = 1M

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

GRID_STAT_ONCE_PER_FIELD = False

MODEL = RTOFS
OBTYPE = GHRSST

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsGHRSST_climWOA_sst

FCST_VAR1_NAME = {CONFIG_DIR}/read_rtofs_ghrsst_woa.py {INPUT_BASE}/model_applications/
↪marine_and_cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst/{valid?fmt=%Y%m%d}_rtofs_
```

(continues on next page)

(continued from previous page)

```
↪glo_2ds_f024_prog.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsGHRSST_climWOA_sst/GHRSST-OSPO-L4-GLOB_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/
↪model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst/OSTIA-
↪UKMO-L4-GLOB-v2.0_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst {valid?fmt=%Y%m%d} fcst

OBS_VAR1_NAME = {CONFIG_DIR}/read_rtofs_ghrsst_woa.py {INPUT_BASE}/model_applications/marine_
↪and_cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst/{valid?fmt=%Y%m%d}_rtofs_glo_2ds_
↪f024_prog.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_
↪obsGHRSST_climWOA_sst/GHRSST-OSPO-L4-GLOB_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_
↪applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst/OSTIA-UKMO-L4-
↪GLOB-v2.0_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪GridStat_fcstRTOFS_obsGHRSST_climWOA_sst {valid?fmt=%Y%m%d} obs

GRID_STAT_CLIMO_MEAN_FIELD = {name="{CONFIG_DIR}/read_rtofs_ghrsst_woa.py {INPUT_BASE}/model_
↪applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst/{valid?fmt=%Y%m
↪%d}_rtofs_glo_2ds_f024_prog.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪GridStat_fcstRTOFS_obsGHRSST_climWOA_sst/GHRSST-OSPO-L4-GLOB_{valid?fmt=%Y%m%d}.nc {INPUT_
↪BASE}/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst/
↪OSTIA-UKMO-L4-GLOB-v2.0_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst {valid?fmt=%Y%m%d} climo"; level="(*,*)
↪";}


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_REGRID_TO_GRID = NONE

GRID_STAT_DESC = NA

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_OUTPUT_PREFIX = SST

GRID_STAT_OUTPUT_FLAG_CNT = BOTH
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
```

(continues on next page)

(continued from previous page)

```
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
```

(continues on next page)

(continued from previous page)

```
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses one Python script to read forecast, observation, and climatology data

parm/use_cases/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst/read_r

```
#!/usr/bin/env python3
"""
Code adapted from
Todd Spindler
NOAA/NWS/NCEP/EMC
Designed to read in RTOFS,GHRSST,WOA and OSTIA data
and based on user input, read sst data
and pass back in memory the forecast, observation, or climatology
data field
"""
```

(continues on next page)

(continued from previous page)

```
import numpy as np
import xarray as xr
import pandas as pd
import pyresample as pyr
from pandas.tseries.offsets import DateOffset
from datetime import datetime, timedelta
from sklearn.metrics import mean_squared_error
import io
from glob import glob
import warnings
import os, sys


if len(sys.argv) < 6:
    print("Must specify the following elements: fcst_file obs_file ice_file, climo_file,␣
↪valid_date, file_flag")
    sys.exit(1)
#grab input files from command line input
rtofsfile = os.path.expandvars(sys.argv[1])
sstfile = os.path.expandvars(sys.argv[2])
icefile = os.path.expandvars(sys.argv[3])
climoDir = os.path.expandvars(sys.argv[4])
vDate=datetime.strptime(sys.argv[5],'%Y%m%d')
file_flag = sys.argv[6]

print('Starting Satellite GHRSST V&V at',datetime.now(),'for',vDate, ' file_flag:',file_flag)

pd.date_range(vDate,vDate)
platform='GHRSST'
param='sst'


######################################################################
# READ GHRSST data ###################################################
######################################################################

if not os.path.exists(sstfile):
        print('missing GHRSST file for',vDate)

sst_data=xr.open_dataset(sstfile,decode_times=True)
sst_data['time']=sst_data.time-pd.Timedelta('12H')  # shift 12Z offset time to 00Z
sst_data2=sst_data.analysed_sst.astype('single')-273.15 # convert from Kelvin
print('Retrieved GHRSST data from NESDIS for',sst_data2.time.values)
```

(continues on next page)

(continued from previous page)

```
sst_data2['lon']=sst_data2.lon.astype('single')
sst_data2['lat']=sst_data2.lat.astype('single')
#sst_data2.attrs['platform']='ghrsst'
sst_data2.attrs['platform']=platform
sst_data2.attrs['units']='degC'

######################################################################
# READ RTOFS data (model output in Tri-polar coordinates) ############
######################################################################

print('reading rtofs ice')
if not os.path.exists(rtofsfile):
    print('missing rtofs file',rtofsfile)
    sys.exit(1)

indata=xr.open_dataset(rtofsfile,decode_times=True)


indata=indata.mean(dim='MT')
indata = indata[param][:-1,]
indata.coords['time']=vDate
#indata.coords['fcst']=fcst

#outdata=indata.copy()
#indata.close()

outdata=indata

outdata=outdata.rename({'Longitude':'lon','Latitude':'lat',})
# all coords need to be single precision
outdata['lon']=outdata.lon.astype('single')
outdata['lat']=outdata.lat.astype('single')
outdata.attrs['platform']='rtofs '+platform

######################################################################
# READ CLIMO WOA data - May require 2 files depending on the date ###
######################################################################

if not os.path.exists(climoDir):
        print('missing climo file file for',vDate)

vDate=pd.Timestamp(vDate)

#climofile="woa13_decav_t{:02n}_04v2.nc".format(vDate.month)
#climo_data=xr.open_dataset(climoDir+'/'+climofile,decode_times=False)
```

(continues on next page)

(continued from previous page)

```
#climo_data=climo_data['t_an'].squeeze()[0,]

if vDate.day==15:  # even for Feb, just because
    climofile="woa13_decav_t{:02n}_04v2.nc".format(vDate.month)
    climo_data=xr.open_dataset(climoDir+'/'+climofile,decode_times=False)
    climo_data=climo_data['t_an'].squeeze()[0,]  # surface only
else:
    if vDate.day < 15:
        start=vDate - DateOffset(months=1,day=15)
        stop=pd.Timestamp(vDate.year,vDate.month,15)
    else:
        start=pd.Timestamp(vDate.year,vDate.month,15)
        stop=vDate + DateOffset(months=1,day=15)
    left=(vDate-start)/(stop-start)

    climofile1="woa13_decav_t{:02n}_04v2.nc".format(start.month)
    climofile2="woa13_decav_t{:02n}_04v2.nc".format(stop.month)
    climo_xr1=xr.open_dataset(climoDir+'/'+climofile1,decode_times=False)
    climo_xr2=xr.open_dataset(climoDir+'/'+climofile2,decode_times=False)
    climo_data1=climo_xr1['t_an'].squeeze()[0,]  # surface only
    climo_data2=climo_xr2['t_an'].squeeze()[0,]  # surface only

    climo_xr1.close()
    climo_xr2.close()

    print('climofile1 :', climofile1)
    print('climofile2 :', climofile2)
    climo_data=climo_data1+((climo_data2-climo_data1)*left)
    climofile='weighted average of '+climofile1+' and '+climofile2

# all coords need to be single precision
climo_data['lon']=climo_data.lon.astype('single')
climo_data['lat']=climo_data.lat.astype('single')
climo_data.attrs['platform']='woa'
climo_data.attrs['filename']=climofile

######################################################################
# READ ICE data for masking ##########################################
######################################################################

if not os.path.exists(icefile):
        print('missing OSTIA ice file for',vDate)

ice_data=xr.open_dataset(icefile,decode_times=True)
ice_data=ice_data.rename({'sea_ice_fraction':'ice'})
```

(continues on next page)

(continued from previous page)

```
# all coords need to be single precision
ice_data2=ice_data.ice.astype('single')
ice_data2['lon']=ice_data2.lon.astype('single')
ice_data2['lat']=ice_data2.lat.astype('single')


def regrid(model,obs):
    """
    regrid data to obs -- this assumes DataArrays
    """
    #model2=model.copy()
    model2=model
    model2_lon=model2.lon.values
    model2_lat=model2.lat.values
    model2_data=model2.to_masked_array()
    if model2_lon.ndim==1:
        model2_lon,model2_lat=np.meshgrid(model2_lon,model2_lat)

    #obs2=obs.copy()
    obs2=obs
    obs2_lon=obs2.lon.astype('single').values
    obs2_lat=obs2.lat.astype('single').values
    obs2_data=obs2.astype('single').to_masked_array()
    if obs2.lon.ndim==1:
        obs2_lon,obs2_lat=np.meshgrid(obs2.lon.values,obs2.lat.values)

    model2_lon1=pyr.utils.wrap_longitudes(model2_lon)
    #model2_lat1=model2_lat.copy()
    model2_lat1=model2_lat
    obs2_lon1=pyr.utils.wrap_longitudes(obs2_lon)
    #obs2_lat1=obs2_lat.copy()
    obs2_lat1=obs2_lat

    # pyresample gausssian-weighted kd-tree interp
    # define the grids
    orig_def = pyr.geometry.GridDefinition(lons=model2_lon1,lats=model2_lat1)
    targ_def = pyr.geometry.GridDefinition(lons=obs2_lon1,lats=obs2_lat1)
    radius=50000
    sigmas=25000
    model2_data2=pyr.kd_tree.resample_gauss(orig_def,model2_data,targ_def,
                                            radius_of_influence=radius,
                                            sigmas=sigmas,
                                            fill_value=None)
    model=xr.DataArray(model2_data2,coords=[obs.lat.values,obs.lon.values],dims=['lat','lon
```

(continues on next page)

(continued from previous page)

```
↪'])

    return model

def expand_grid(data):
    """
    concatenate global data for edge wraps
    """

    data2=data.copy()
    data2['lon']=data2.lon+360
    data3=xr.concat((data,data2),dim='lon')
    data2.close()
    data.close()
    return data3

sst_data2=sst_data2.squeeze()

#print('regridding climo to obs')
climo_data=climo_data.squeeze()
climo_data=regrid(climo_data,sst_data2)

#print('regridding ice to obs')
ice_data2=regrid(ice_data2,sst_data2)

#print('regridding model to obs')
model2=regrid(outdata,sst_data2)

# combine obs ice mask with ncep
obs2=sst_data2.to_masked_array()
ice2=ice_data2.to_masked_array()
climo2=climo_data.to_masked_array()
model2=model2.to_masked_array()

#reconcile with obs
obs2.mask=np.ma.mask_or(obs2.mask,ice2>0.0)
obs2.mask=np.ma.mask_or(obs2.mask,climo2.mask)
obs2.mask=np.ma.mask_or(obs2.mask,model2.mask)
climo2.mask=obs2.mask
model2.mask=obs2.mask

coord_lat = sst_data2.lat.values
coord_lon = sst_data2.lon.values

sst_data2.close()
```

(continues on next page)

(continued from previous page)

```
#Create the MET grids based on the file_flag
if file_flag == 'fcst':
    #model2=xr.DataArray(model2,coords=[sst_data2.lat.values,sst_data2.lon.values], dims=[
↪'lat','lon'])
    model2=xr.DataArray(model2,coords=[coord_lat,coord_lon], dims=['lat','lon'])
    model2=expand_grid(model2)
    met_data = model2[:,:]
    #trim the lat/lon grids so they match the data fields
    lat_met = model2.lat
    lon_met = model2.lon
    #print(" RTOFS Data shape: "+repr(met_data.shape))
    v_str = vDate.strftime("%Y%m%d")
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'sst',
            'standard_name': 'sst',
            'long_name': 'sst',
            'level': "SURFACE",
            'units': "degC",

            'grid': {
                'type': "LatLon",
                'name': "RTOFS Grid",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
```

(continues on next page)

(continued from previous page)

```
    attrs = met_data.attrs

if file_flag == 'obs':
    #obs2=xr.DataArray(obs2,coords=[sst_data2.lat.values,sst_data2.lon.values], dims=['lat',
↪'lon'])
    obs2=xr.DataArray(obs2,coords=[coord_lat, coord_lon], dims=['lat','lon'])
    obs2=expand_grid(obs2)
    met_data = obs2[:,:]
    #trim the lat/lon grids so they match the data fields
    lat_met = obs2.lat
    lon_met = obs2.lon
    v_str = vDate.strftime("%Y%m%d")
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'sst',
            'standard_name': 'analyzed sst',
            'long_name': 'analyzed sst',
            'level': "SURFACE",
            'units': "degC",

            'grid': {
                'type': "LatLon",
                'name': "Lat Lon",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs
```

(continues on next page)

(continued from previous page)

```
if file_flag == 'climo':
    #climo2=xr.DataArray(climo2,coords=[sst_data2.lat.values,sst_data2.lon.values], dims=[
↪'lat','lon'])
    climo2=xr.DataArray(climo2,coords=[coord_lat, coord_lon], dims=['lat','lon'])
    climo2=expand_grid(climo2)
    met_data = climo2[:,:]
    #modify the lat and lon grids since they need to match the data dimensions, and code␣
↪cuts the last row/column of data
    lat_met = climo2.lat
    lon_met = climo2.lon
    v_str = vDate.strftime("%Y%m%d")
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'sea_water_temperature',
            'standard_name': 'sea_water_temperature',
            'long_name': 'sea_water_temperature',
            'level': "SURFACE",
            'units': "degC",

            'grid': {
                'type': "LatLon",
                'name': "crs Grid",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstRTOFS_obsGHRSST_climWOA_sst.conf then a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/marine_and_cryosphere/
↪GridStat_fcstRTOFS_obsGHRSST_climWOA_sst.conf /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstRTOFS_obsGHRSST_climWOA_sst.conf:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/marine_and_cryosphere/
↪GridStat_fcstRTOFS_obsGHRSST_climWOA_sst.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[config]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for thisIce use case will be found in directory 20210503 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_SST_000000L_20210503_000000V.stat
- grid_stat_SST_000000L_20210503_000000V_cnt.txt
- grid_stat_SST_000000L_20210503_000000V_pairs.nc

Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase
- ClimatologyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/marine_and_cryosphere-GridStat_fcstRTOFS_obsGHRSST_climWOA_sst.pn

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.6.4 GridStat: Python Embedding to read and process sea surface heights

model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh.conf

Scientific Objective

This use case utilizes Python embedding to extract several statistics from the sea surface height data over the globe, which was already being done in a closed system. By producing the same output via METplus, this use case provides standardization and reproducible results.

Datasets

Forecast: RTOFS ssh file via Python Embedding script/file

Observations: AVISO ssh file via Python Embedding script/file

Sea Ice Masking: RTOFS ice cover file via Python Embedding script/file

Climatology: HYCOM ssh file via Python Embedding script/file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1132) section for more information.

Data Source: COPERNICUS GLOBAL OCEAN SSH NRT (LEVEL 4), HYCOM + NCODA Global 1/12 deg Reanalysis

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- scikit-learn
- pyresample

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case utilizes the METplus GridStat wrapper to generate a command to run the MET tool GridStat with Python Embedding for the specified user hemispheres

METplus Workflow

GridStat is the only tool called in this example. This use case will pass in both the observation, forecast, and climatology gridded data being pulled from the files via Python Embedding. All of the desired statistics reside in the CNT line type, so that is the only output requested. It processes the following run time:

Valid: 2021-05-11 0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG=20210811
VALID_END=20210811
VALID_INCREMENT = 1M

LEAD_SEQ = 24


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE = PYTHON_NUMPY
GRID_STAT_CLIMO_MEAN_FIELD = {name="{CONFIG_DIR}/read_rtofs_aviso_hycom.py {INPUT_BASE}/
↪model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh/{init?
↪fmt=%Y%m%d}_rtofs_glo_2ds_f024_diag.nc {INPUT_BASE}/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh/nrt_global_allsat_phy_l4_{valid?fmt=%Y
↪%m%d}.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsAVISO_
↪climHYCOM_ssh/OSTIA-UKMO-L4-GLOB-v2.0_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_
↪applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh {valid?fmt=%Y
↪%m%d} climo"; level="(*,*)";}

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

GRID_STAT_ONCE_PER_FIELD = False

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsAVISO_climHYCOM_ssh

FCST_VAR1_NAME = {CONFIG_DIR}/read_rtofs_aviso_hycom.py {INPUT_BASE}/model_applications/
↪marine_and_cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh/{init?fmt=%Y%m%d}_rtofs_
↪glo_2ds_f024_diag.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsAVISO_climHYCOM_ssh/nrt_global_allsat_phy_l4_{valid?fmt=%Y%m%d}.nc {INPUT_
↪BASE}/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh/
↪OSTIA-UKMO-L4-GLOB-v2.0_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh {valid?fmt=%Y%m%d} fcst

OBS_VAR1_NAME = {CONFIG_DIR}/read_rtofs_aviso_hycom.py {INPUT_BASE}/model_applications/
↪marine_and_cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh/{init?fmt=%Y%m%d}_rtofs_
↪glo_2ds_f024_diag.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsAVISO_climHYCOM_ssh/nrt_global_allsat_phy_l4_{valid?fmt=%Y%m%d}.nc {INPUT_
↪BASE}/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh/
↪OSTIA-UKMO-L4-GLOB-v2.0_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/marine_and_
```

(continues on next page)

(continued from previous page)

```
↪cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh {valid?fmt=%Y%m%d} obs


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_REGRID_TO_GRID = NONE

MODEL = RTOFS

OBTYPE = AVISO

GRID_STAT_DESC = NA

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_OUTPUT_PREFIX = SSH

GRID_STAT_OUTPUT_FLAG_CNT = BOTH
GRID_STAT_OUTPUT_FLAG_SAL1L2 = BOTH
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
```

(continues on next page)

(continued from previous page)

```
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
```

(continues on next page)

(continued from previous page)

```
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses one Python script to read forecast and observation data

parm/use_cases/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh/read_

```
#!/bin/env python
"""
Code adapted from
Todd Spindler
NOAA/NWS/NCEP/EMC
Designed to read in RTOFS,AVISO,HYCOM and OSTIA data
and based on user input, read ssh data
and pass back in memory the forecast, observation, or climatology
data field
"""

import numpy as np
import xarray as xr
import pandas as pd
import pyresample as pyr
from pandas.tseries.offsets import DateOffset
from datetime import datetime, timedelta
from sklearn.metrics import mean_squared_error
import io
from glob import glob
import warnings
import os, sys

if len(sys.argv) < 6:
    print("Must specify the following elements: fcst_file obs_file ice_file, climo_file,␣
↪valid_date, file_flag")
    sys.exit(1)
```

(continues on next page)

(continued from previous page)

```
rtofsfile = os.path.expandvars(sys.argv[1])
sshfile = os.path.expandvars(sys.argv[2])
icefile = os.path.expandvars(sys.argv[3])
climoDir = os.path.expandvars(sys.argv[4])
vDate=datetime.strptime(sys.argv[5],'%Y%m%d')
file_flag = sys.argv[6]

print('Starting Satellite AVISO V&V at',datetime.now(),'for',vDate, ' file_flag:',file_flag)

pd.date_range(vDate,vDate)
platform='AVISO'
param='ssh'


######################################################################
# READ AVISO data ####################################################
######################################################################

if not os.path.exists(sshfile):
        print('missing AVISO file for',vDate)

ssh_data=xr.open_dataset(sshfile,decode_times=True)
print('Retrieved SSH above sea level AVISO data from NESDIS for',ssh_data.time.values)
sla=ssh_data.sla.astype('single')
sla.attrs['platform']=platform
sla.attrs['time']=pd.Timestamp(ssh_data.time.values[0])
sla=sla.rename({'longitude':'lon','latitude':'lat'})
sla.attrs['filename']=sshfile.split('/')[-1]


# all coords need to be single precision
sla['lon']=sla.lon.astype('single')
sla['lat']=sla.lat.astype('single')
sla.attrs['units']='meters'

adt=ssh_data.adt.astype('single')
adt.attrs['platform']='aviso'
adt.attrs['filename']=sshfile
adt.attrs['time']=pd.Timestamp(ssh_data.time.values[0])
adt=adt.rename({'longitude':'lon','latitude':'lat'})
# all coords need to be single precision
adt['lon']=adt.lon.astype('single')
adt['lat']=adt.lat.astype('single')
adt.attrs['units']='meters'
```

(continues on next page)

(continued from previous page)

```
sla=sla.squeeze()
adt=adt.squeeze()

######################################################################
# READ RTOFS data (model output in Tri-polar coordinates) ############
######################################################################

print('reading rtofs ice')
if not os.path.exists(rtofsfile):
    print('missing rtofs file',rtofsfile)
    sys.exit(1)

indata=xr.open_dataset(rtofsfile,decode_times=True)


indata=indata.mean(dim='MT')
indata = indata[param][:-1,]
indata.coords['time']=vDate
#indata.coords['fcst']=fcst

outdata=indata.copy()

outdata=outdata.rename({'Longitude':'lon','Latitude':'lat',})
# all coords need to be single precision
outdata['lon']=outdata.lon.astype('single')
outdata['lat']=outdata.lat.astype('single')
outdata.attrs['platform']='rtofs '+platform

######################################################################
# READ CLIMO HYCOM data - May require 2 files depending on the date ###
######################################################################

if not os.path.exists(climoDir):
        print('missing climo file file for',vDate)

vDate=pd.Timestamp(vDate)

climofile="hycom_GLBv0.08_53X_archMN.1994_{0:02n}_2015_{0:02n}_ssh.nc".format(vDate.month)
climo_data=xr.open_dataset(climoDir+'/'+climofile,decode_times=False)

if vDate.day==15:  # even for Feb, just because
    climofile="hycom_GLBv0.08_53X_archMN.1994_{0:02n}_2015_{0:02n}_ssh.nc".format(vDate.
↪month)
    climo_data=xr.open_dataset(climoDir+'/'+climofile,decode_times=False)
```

(continues on next page)

(continued from previous page)

```
    climo_data=climo_data['surf_el'].copy().squeeze()
else:
    if vDate.day < 15:
        start=vDate - DateOffset(months=1,day=15)
        stop=pd.Timestamp(vDate.year,vDate.month,15)
    else:
        start=pd.Timestamp(vDate.year,vDate.month,15)
        stop=vDate + DateOffset(months=1,day=15)
    left=(vDate-start)/(stop-start)

    climofile1="hycom_GLBv0.08_53X_archMN.1994_{0:02n}_2015_{0:02n}_ssh.nc".format(start.
↪month)
    climofile2="hycom_GLBv0.08_53X_archMN.1994_{0:02n}_2015_{0:02n}_ssh.nc".format(stop.
↪month)
    climo_data1=xr.open_dataset(climoDir+'/'+climofile1,decode_times=False)
    climo_data2=xr.open_dataset(climoDir+'/'+climofile2,decode_times=False)
    climo_data1=climo_data1['surf_el'].copy().squeeze()
    climo_data2=climo_data2['surf_el'].copy().squeeze()
    climo_data=climo_data1+((climo_data2-climo_data1)*left)
    climofile='weighted average of '+climofile1+' and '+climofile2

    print('climofile1 :', climofile1)
    print('climofile2 :', climofile2)

climo_data.coords['time']=datetime(vDate.year,vDate.month,1)   # just a reference to the␣
↪month
# all coords need to be single precision

climo_data['lon']=climo_data.lon.astype('single')
climo_data['lat']=climo_data.lat.astype('single')
climo_data.attrs['platform']='hycom'
climo_data.attrs['filename']=climofile

#########################################################################
# READ ICE data for masking ##############################################
#########################################################################

if not os.path.exists(icefile):
        print('missing OSTIA ice file for',vDate)

ice_data=xr.open_dataset(icefile,decode_times=True)
ice_data=ice_data.rename({'sea_ice_fraction':'ice'})

# all coords need to be single precision
ice_data2=ice_data.ice.astype('single')
```

(continues on next page)

(continued from previous page)

```
ice_data2['lon']=ice_data2.lon.astype('single')
ice_data2['lat']=ice_data2.lat.astype('single')




def regrid(model,obs):
    """
    regrid data to obs -- this assumes DataArrays
    """
    model2=model.copy()
    model2_lon=model2.lon.values
    model2_lat=model2.lat.values
    model2_data=model2.to_masked_array()
    if model2_lon.ndim==1:
        model2_lon,model2_lat=np.meshgrid(model2_lon,model2_lat)

    obs2=obs.copy()
    obs2_lon=obs2.lon.astype('single').values
    obs2_lat=obs2.lat.astype('single').values
    obs2_data=obs2.astype('single').to_masked_array()
    if obs2.lon.ndim==1:
        obs2_lon,obs2_lat=np.meshgrid(obs2.lon.values,obs2.lat.values)

    model2_lon1=pyr.utils.wrap_longitudes(model2_lon)
    model2_lat1=model2_lat.copy()
    obs2_lon1=pyr.utils.wrap_longitudes(obs2_lon)
    obs2_lat1=obs2_lat.copy()

    # pyresample gausshian-weighted kd-tree interp
    # define the grids
    orig_def = pyr.geometry.GridDefinition(lons=model2_lon1,lats=model2_lat1)
    targ_def = pyr.geometry.GridDefinition(lons=obs2_lon1,lats=obs2_lat1)
    radius=50000
    sigmas=25000
    model2_data2=pyr.kd_tree.resample_gauss(orig_def,model2_data,targ_def,
                                            radius_of_influence=radius,
                                            sigmas=sigmas,
                                            fill_value=None)
    model=xr.DataArray(model2_data2,coords=[obs.lat.values,obs.lon.values],dims=['lat','lon
↪'])

    return model

def expand_grid(data):
```

(continues on next page)

(continued from previous page)

```
    """
    concatenate global data for edge wraps
    """

    data2=data.copy()
    data2['lon']=data2.lon+360
    data3=xr.concat((data,data2),dim='lon')
    return data3

print('regridding climo to obs')
climo_data=climo_data.squeeze()
climo_data=regrid(climo_data,adt)

print('regridding ice to obs')
ice_data2=regrid(ice_data2,adt)

print('regridding model to obs')
model2=regrid(outdata,adt)

# combine obs ice mask with ncep
obs2=adt.to_masked_array()
obs_anom=sla.copy()
obs_anom2=obs_anom.to_masked_array()
ice2=ice_data2.to_masked_array()
climo2=climo_data.to_masked_array()
model2=model2.to_masked_array()

#reconcile with obs
obs2.mask=np.ma.mask_or(obs2.mask,ice2>0.0)
obs2.mask=np.ma.mask_or(obs2.mask,climo2.mask)
obs2.mask=np.ma.mask_or(obs2.mask,model2.mask)
climo2.mask=obs2.mask
model2.mask=obs2.mask
obs_anom2.mask=obs2.mask

obs2=xr.DataArray(obs2,coords=[adt.lat.values,adt.lon.values], dims=['lat','lon'])
obs_anom2=xr.DataArray(obs_anom2,coords=[adt.lat.values,adt.lon.values], dims=['lat','lon'])
model2=xr.DataArray(model2,coords=[adt.lat.values,adt.lon.values], dims=['lat','lon'])
climo2=xr.DataArray(climo2,coords=[adt.lat.values,adt.lon.values], dims=['lat','lon'])

model2=expand_grid(model2)
climo2=expand_grid(climo2)
obs2=expand_grid(obs2)
obs_anom2=expand_grid(obs_anom2)
```

(continues on next page)

(continued from previous page)

```
#Modify the lat/lon min/max values to subset the data
model3=model2.where((model2.lon>=0)&(model2.lon<=360)&
        (model2.lat>=-80)&(model2.lat<=90),drop=True)
climo3=climo2.where((climo2.lon>=0)&(climo2.lon<=360)&
        (climo2.lat>=-80)&(climo2.lat<=90),drop=True)
obs3=obs2.where((obs2.lon>=0)&(obs2.lon<=360)&
        (obs2.lat>=-80)&(obs2.lat<=90),drop=True)
obs_anom3=obs_anom2.where((obs_anom2.lon>=0)&(obs_anom2.lon<=360)&
        (obs_anom2.lat>=-80)&(obs_anom2.lat<=90),drop=True)

#Create the MET grids based on the file_flag
if file_flag == 'fcst':
    met_data = model3[:,:]
    #trim the lat/lon grids so they match the data fields
    lat_met = model3.lat
    lon_met = model3.lon
    print(" RTOFS Data shape: "+repr(met_data.shape))
    v_str = vDate.strftime("%Y%m%d")
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'ssh',
            'standard_name': 'sea_surface_elevation',
            'long_name': 'sea surf. height  [92.8H]',
            'level': "SURFACE",
            'units': "meters",

            'grid': {
                'type': "LatLon",
                'name': "RTOFS Grid",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
```

(continues on next page)

(continued from previous page)

```
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs

if file_flag == 'obs':
    met_data = obs3[:,:]
    #trim the lat/lon grids so they match the data fields
    lat_met = obs3.lat
    lon_met = obs3.lon
    v_str = vDate.strftime("%Y%m%d")
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'ssh',
            'standard_name': 'sea_surface_height_above_geoid',
            'long_name': 'absolute_dynamic_topography',
            'level': "SURFACE",
            'units': "meters",

            'grid': {
                'type': "LatLon",
                'name': "Lat Lon",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
```

(continues on next page)

(continued from previous page)

```
    attrs = met_data.attrs

if file_flag == 'climo':
    met_data = climo3[:,:]
    #modify the lat and lon grids since they need to match the data dimensions, and code␣
↪cuts the last row/column of data
    lat_met = climo3.lat
    lon_met = climo3.lon
    v_str = vDate.strftime("%Y%m%d")
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'sea_surface_height',
            'standard_name': 'sea_surface_elevation',
            'long_name': 'Water Surface Elevation',
            'level': "SURFACE",
            'units': "meters",

            'grid': {
                'type': "LatLon",
                'name': "crs Grid",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for thisIce use case will be found in 20210503 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_SSH_000000L_20210811_000000V.stat
- grid_stat_SSH_000000L_20210811_000000V_sal1l2.txt
- grid_stat_SSH_000000L_20210811_000000V_cnt.txt

- grid_stat_SSH_000000L_20210811_000000V_pairs.nc

Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase
- ClimatologyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/marine_and_cryosphere-GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh.pr

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.6.5 UserScript: Python Script to compute cable transport

model_applications/marine_and_cryosphere/UserScript_fcstRTOFS_obsAOML_calcTransport.conf

Scientific Objective

The Florida Current flows northward along the eastern Florida coast and feeds to the Gulf Stream. More info can be obtained from: https://www.aoml.noaa.gov/phod/floridacurrent/index.php

This use case utilizes a Python script to calculate transport (units Sv) variations of the Florida current using a submarine cable and snapshot estimates made by shipboard instruments. The code compares the transport using RTOFS data and compare it with the AOML cable transport data and computes BIAS, RMSE, CORRELATION, and Scatter Index. The operational code utilizes 21 days of data and computes 7 day statistics. For the use case 3 days of data are utilized. The valid date is passed though an argument. The valid date is the last processed day i.e. the code grabs 3 previous days of data.

Datasets

Forecast: RTOFS u(3zuio) amd ,v(3zvio) files via Python Embedding script/file

Observations: AOML Florida Current data via Python Embedding script/file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1141) section for more information.

Data Source: NOMADS RTOFS Global + Daily mean transport (https://www.aoml.noaa.gov/phod/floridacurrent/data_access.php)+ Eightmilecable (static, provided with the use case)

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- scikit-learn
- pyproj

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

```
[user_env_vars]
MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python
```

METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command to run with Python Embedding for the specified valid time.

METplus Workflow

This use case uses UserScript. All the gridded data being pulled from the files via Python Embedding. All of the desired statistics are in the log file. It processes the following run time:

Valid: 2021-10-28

The code grabs the 20211028, 20211027, and 20211026 24 hour RTOFS files.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. -c parm/use_cases/model_applications/marine_and_cryosphere/UserScript_fcstRTOFS_obsAOML_calcTransport.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/marine_and_
↪cryosphere/UserScript_fcstRTOFS_obsAOML_calcTransport.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript

###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20211028
VALID_INCREMENT = 24H

LEAD_SEQ =

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
```

(continues on next page)

(continued from previous page)

```
###

USER_SCRIPT_INPUT_TEMPLATE = {VALID_BEG}

USER_SCRIPT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/marine_and_cryosphere/calc_
↪transport

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/
↪UserScript_fcstRTOFS_obsAOML_calcTransport/read_aomlcable_rtofs_transport.py {USER_SCRIPT_
↪INPUT_TEMPLATE}


[user_env_vars]

# Calc Transport specific variables

CALC_TRANSPORT_RTOFS_DIRNAME = {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪UserScript_fcstRTOFS_obsAOML_calcTransport/RTOFS

CALC_TRANSPORT_CABLE_FILENAME = {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪UserScript_fcstRTOFS_obsAOML_calcTransport/FC_cable_transport_2021.dat

CALC_TRANSPORT_EIGHTMILE_FILENAME = {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪UserScript_fcstRTOFS_obsAOML_calcTransport/eightmilecable.dat

CALC_TRANSPORT_LEAD_TIME = 24

# Calculate stats for number of days. The operational website uses 21 days
# of data and then calculates 7 day stats. For the use case both of them are 3 days each.
# The code calculates the number of subdirectories
# under RTOFS directory, however, CALC_TRANSPORT_STATS_DAY is the number of days the␣
↪statistics
# will be calculated.
CALC_TRANSPORT_STATS_DAY = 3

CALC_TRANSPORT_LOG_FILE = calc_transport.log

OUTPUT_DIR = {USER_SCRIPT_OUTPUT_DIR}
```

MET Configuration

None. All of the processing is completed in the UserScript

User Script

This use case uses one Python script to read forecast and observation data as well as processing the desired statistics.

parm/use_cases/model_applications/marine_and_cryosphere/UserScript_fcstRTOFS_obsAOML_calcTransport/read_a

```
#! /usr/bin/env python3
"""
Florida Cable Transport Class-4 Validation System
Adapted from Todd Spindler's code
"""

from netCDF4 import Dataset
import numpy as np
from pyproj import Geod
import math
from sklearn.metrics import mean_squared_error
from datetime import datetime, timedelta
import pandas as pd
import sys, os
import logging

vDate=datetime.strptime(sys.argv[1],'%Y%m%d')
rtofsdir = os.environ.get('CALC_TRANSPORT_RTOFS_DIRNAME')
cablefile = os.environ.get('CALC_TRANSPORT_CABLE_FILENAME')
eightmilefile = os.environ.get('CALC_TRANSPORT_EIGHTMILE_FILENAME')

print('Starting Cable V&V at',datetime.now(),'for',vDate)



if not os.path.exists(cablefile):
        print('missing AOML Cable transport file for',vDate)

#-----------------------------------------------
# read cable transport data from AOML
#-----------------------------------------------

# read the AOML dataset
names=['year','month','day','transport']
cable=pd.read_csv(cablefile,comment='%',names=names,delimiter=' ',
```

(continues on next page)

(continued from previous page)

```
    skipinitialspace=True,header=None,usecols=list(range(4)))
cable['date']=pd.to_datetime(cable[['year','month','day']])
cable.index=cable.date
cable['error']=2.0
del cable['year'], cable['month'], cable['day'], cable['date']
print(cable)

#-----------------------------------------------
# full cross-section transport calculation
#-----------------------------------------------
def calc_transport(dates,fcst):
    """
    Calculate the transport of water across the Florida Straits
    This extracts the section and integrates the flow through it.
    """
    transport=[]
    fcst_str='f{:03d}'.format(fcst)
    cable_loc=np.loadtxt(eightmilefile,dtype='int',usecols=(0,1))
    eightmile_lat = 26.5167
    eightmile_lon = -78.7833%360
    wpb_lat = 26.7153425
    wpb_lon = -80.0533746%360
    cable_angle = math.atan((eightmile_lat-wpb_lat)/(eightmile_lon-wpb_lon))
    g=Geod(ellps='WGS84')

    for date in dates:
        print('DATE :', date, ' DATES :',dates)
        print('processing',date.strftime('%Y%m%d'),'fcst',fcst)
        rundate=date-timedelta(fcst/24.)  # calc rundate from fcst and date
        ufile=rtofsdir+'/'+rundate.strftime('%Y%m%d')+'/rtofs_glo_3dz_'+fcst_str+'_daily_
↪3zuio.nc'
        vfile=rtofsdir+'/'+rundate.strftime('%Y%m%d')+'/rtofs_glo_3dz_'+fcst_str+'_daily_
↪3zvio.nc'

        print(ufile)
        print(vfile)

        udata=Dataset(ufile)
        vdata=Dataset(vfile)

        lon=udata['Longitude'][:]
        lat=udata['Latitude'][:]
        depth=udata['Depth'][:]

        usection=np.zeros((depth.shape[0],cable_loc.shape[0]))
```

(continues on next page)

(continued from previous page)

```
        vsection=np.zeros((depth.shape[0],cable_loc.shape[0]))

        udata=udata['u'][:].squeeze()
        vdata=vdata['v'][:].squeeze()

        for ncol,(row,col) in enumerate(cable_loc):
            usection[:,ncol]=udata[:,row,col].filled(fill_value=0.0)
            vsection[:,ncol]=vdata[:,row,col].filled(fill_value=0.0)

        lon=lon[cable_loc[:,0],cable_loc[:,1]]
        lat=lat[cable_loc[:,0],cable_loc[:,1]]

        # compute the distances along the track
        _,_,dist=g.inv(lon[0:-1],lat[0:-1],lon[1:],lat[1:])
        depth=np.diff(depth)
        usection=usection[:-1,:-1]
        vsection=vsection[:-1,:-1]

        dist,depth=np.meshgrid(dist,depth)
        u,v=rotate(usection,vsection,cable_angle)
        trans1=(v*dist*depth).sum()/1e6
        #print(date.strftime('%Y-%m-%d'),' transport:',transport,'Sv')
        transport.append(trans1)

    return transport

#-----------------------------------------------
# retrieve model data
#-----------------------------------------------
def get_model(dates,fcsts):

    transport={'dates':dates}


    for fcst in fcsts:
            transport[fcst]=calc_transport(dates,fcst)

    model=pd.DataFrame(transport)
    model.index=model.dates
    del model['dates']
    #del model['validDates']

    print(model)
    return model
#-----------------------------------------------
```

(continues on next page)

(continued from previous page)

```
# coordinate rotation
#-----------------------------------------------
def rotate(u,v,phi):
    # phi is in radians
    u2 =  u*math.cos(phi) + v*math.sin(phi)
    v2 = -u*math.sin(phi) + v*math.cos(phi)
    return u2,v2

#-----------------------------------------------
if __name__ == "__main__":

    want_date=vDate
    DateSet=True

    fcst = int(os.environ.get('CALC_TRANSPORT_LEAD_TIME'))
    no_of_fcst_stat_days = int(os.environ.get('CALC_TRANSPORT_STATS_DAY'))

    fcsts=list(range(fcst,fcst+1,24))

    start_date=want_date
    stop_date=want_date
    cable=cable[:stop_date]

    # Count the number in the subdirs RTOFS dir
    path, dirs, files = next(os.walk(rtofsdir))
    dir_count = len(dirs)
    dir_count

    """
    Setup logging
    """
    logfile = os.environ.get('CALC_TRANSPORT_LOG_FILE')


    for end_date in pd.date_range(start_date,stop_date):
        dates=pd.date_range(end=end_date,periods=dir_count)
        model=get_model(dates,fcsts)

    both=pd.merge(cable,model,left_index=True,right_index=True,how='inner')
    print("both :", both)
    both=both[both.index.max()-timedelta(no_of_fcst_stat_days):]

    diff=both[fcst] - both.transport
    bias=diff.mean()
    rmse=mean_squared_error(both.transport,both[fcst])**0.5
```

(continues on next page)

(continued from previous page)

```
    if both[fcst].mean() != 0.0:
        scatter_index=100.0*(((diff**2).mean())**0.5 - bias**2)/both.transport.mean()
    else:
        scatter_index=np.nan

    corr=both[fcst].corr(both.transport)

#    print("BIAS :",bias, "RMSE :",rmse, "CORR :",corr, "SCATTER INDEX :",scatter_index)

    outdir = os.environ.get('OUTPUT_DIR')

    if not os.path.exists(outdir):
        print(f"Creating output directory: {outdir}")
        os.makedirs(outdir)

    expected_file = os.path.join(outdir,logfile)
    print(expected_file)

    with open(expected_file, 'w') as f:
       print("BIAS :",bias, "RMSE :",rmse, "CORR :",corr, "SCATTER INDEX :",scatter_index,␣
↪file=f)
```

Running METplus

This use case can be run two ways:

1) Passing in UserScript_fcstRTOFS_obsAOML_calcTransport.conf then a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/marine_and_cryosphere/
↪UserScript_fcstRTOFS_obsAOML_calcTransport.conf /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_fcstRTOFS_obsAOML_calcTransport.conf:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/marine_and_cryosphere/
↪UserScript_fcstRTOFS_obsAOML_calcTransport.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[config]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for use case will be found in calc_transport (relative to **OUTPUT_BASE**) and will contain the following files:

- calc_transport.log

Keywords

Note:

- UserScriptUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/marine_and_cryosphere-UserScript_fcstRTOFS_obsAOML_calcTransport.png

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.6.6 PlotDataPlane: Python Embedding of tripolar coordinate file

model_applications/marine_and_cryosphere/PlotDataPlane_obsHYCOM_coordTripolar.conf

Scientific Objective

By producing a postscript image from a file that utilizes a tripolar coordinate system, this use case shows METplus can utilize python embedding to ingest and utilize file structures on the same coordinate system.

Datasets

Input: Python Embedding script/file, HYCOM observation file, coordinate system weight files (optional)

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1150) section for more information.

Data Source: HYCOM model

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- xesmf

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case utilizes the METplus PlotDataPlane wrapper to generate a command to run the MET tool PlotDataPlane with Python Embedding if all required files are found.

METplus Workflow

PlotDataPlane is the only tool called in this example. It processes the following run time:

Valid: 2020-01-27 0Z

As it is currently set, the configuration file will pass in the path to the observation data, as well as a path to the weights for the coordinate system. This is done in an effort to speed up running the use case. These weight files are not required to run at the time of executing the use case, but will be made via Python Embedding if they are not found/passed in at run time. Additional user configurations, including the lat/lon spacing, can be found in the python script.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/marine_and_cryosphere/PlotDataPlane_obsHYCOM_coordTripolar.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/marine_and_
↪cryosphere/PlotDataPlane_obsHYCOM_coordTripolar.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PlotDataPlane


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20200127
VALID_END = 20200127
VALID_INCREMENT = 1M

LEAD_SEQ = 0

PLOT_DATA_PLANE_CUSTOM_LOOP_LIST = north, south


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

PLOT_DATA_PLANE_INPUT_TEMPLATE = PYTHON_NUMPY

PLOT_DATA_PLANE_OUTPUT_TEMPLATE = {OUTPUT_BASE}/model_applications/marine_and_cryosphere/
↪PlotDataPlane_obsHYCOM_coordTripolar/HYCOM_iceCoverage_{custom}.ps


###
# PlotDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#plotdataplane
###

LOG_PLOT_DATA_PLANE_VERBOSITY = 1

PLOT_DATA_PLANE_FIELD_NAME = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/
↪PlotDataPlane_obsHYCOM_coordTripolar/read_tripolar_grid.py {INPUT_BASE}/model_applications/
↪marine_and_cryosphere/PlotDataPlane_obsHYCOM_coordTripolar/rtofs_glo_2ds_n048_daily_diag.
↪nc ice_coverage {custom} {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪PlotDataPlane_obsHYCOM_coordTripolar/weight_{custom}.nc

PLOT_DATA_PLANE_TITLE = Tripolar via Python

PLOT_DATA_PLANE_COLOR_TABLE =

PLOT_DATA_PLANE_RANGE_MIN_MAX =
```

MET Configuration

This tool does not use a MET configuration file.

Python Embedding

This use case uses one Python script to read input data, passed through two times

parm/use_cases/model_applications/marine_and_cryosphere/PlotDataPlane_obsHYCOM_coordTripolar/read_tripolar

```
import os
import sys
import pandas as pd
import xarray as xr
import xesmf as xe

################################################################################
# This script reads in tripolar grid ice data from the rtofs model and
# passes it to MET tools through python embedding.
# Written by George McCabe, NCAR
# January 2021
# Python embedding structure adapted from read_PostProcessed_WRF.py from
# the DTC MET User's Page.
# Tripolar grid logic adapted from ice_cover.py
# from Todd Spindler, NOAA/NCEP/EMC.
# Based on a script written by Lindsay Blank, NCAR in April 2020
# Arguments:
#  input filename - path to input NetCDF file to process
#  field name - name of field to read (ice_coverage or ice thickness)
#  hemisphere - hemisphere to process (north or south)
# Example call: read_tripolar_grid.py /path/to/file.nc ice_coverage north
################################################################################


# degrees between lat/lon points in output grid
LATITUDE_SPACING = 0.25
LONGITUDE_SPACING = 0.25

# set DEBUG to True to get debugging output
DEBUG = False

# latitude boundaries where curved data begins
# we are only concerned with data outside of the boundary for this case
# so we crop data that is below (for north) or above (for south)
LAT_BOUND_NORTH = 30.98
LAT_BOUND_SOUTH = -39.23
```

(continues on next page)

(continued from previous page)

```
# list of valid values to specify for hemisphere
HEMISPHERES = ['north', 'south']

def print_min_max(ds):
    print(f"MIN LAT: {float(ds['lat'].min())} and "
          f"MIN LON: {float(ds['lon'].min())}")
    print(f"MAX LAT: {float(ds['lat'].max())} and "
          f"MAX LON: {float(ds['lon'].max())}")

if len(sys.argv) < 4:
    print("Must specify exactly one input file and variable name.")
    sys.exit(1)

# Read the input file as the first argument
input_file = os.path.expandvars(sys.argv[1])
var = sys.argv[2]
hemisphere = sys.argv[3]

# read optional weight file if provided
if len(sys.argv) > 4:
    weight_file = sys.argv[4]
else:
    weight_file = f'weight_{hemisphere}.nc'

if hemisphere not in HEMISPHERES:
    print(f"ERROR: Invalid hemisphere value ({hemisphere}) "
          f"Valid options are {HEMISPHERES}")
    sys.exit(1)

try:
    # Print some output to verify that this script ran
    print(f"Input File: {repr(input_file)}")
    print(f"Variable: {repr(var)}")
    print(f"Hemisphere: {repr(hemisphere)}")

    # read input file
    xr_dataset = xr.load_dataset(input_file,
                                 decode_times=True)
except NameError:
    print("Trouble reading data from input file")
    sys.exit(1)

# get time information
```

(continues on next page)

(continued from previous page)

```
dt = pd.to_datetime(str(xr_dataset.MT[0].values))
valid_time = dt.strftime('%Y%m%d_%H%M%S')

# rename Latitude and Longitude to format that xesmf expects
xr_dataset = xr_dataset.rename({'Longitude': 'lon', 'Latitude': 'lat'})
# drop singleton time dimension for this example
xr_dataset = xr_dataset.squeeze()

# print out input data for debugging
if DEBUG:
    print("INPUT DATASET:")
    print(xr_dataset)
    print_min_max(xr_dataset)
    print('\n\n')

# get field name values to read into attrs
standard_name = xr_dataset[var].standard_name
long_name = xr_dataset[var].long_name.strip()

# trim off row of data
xr_dataset = xr_dataset.isel(Y=slice(0,-1))

# remove data inside boundary latitude to get only curved data
if hemisphere == 'north':
    xr_out_bounds = xr_dataset.where(xr_dataset.lat >= LAT_BOUND_NORTH,
                                     drop=True)
    lat_min = xr_out_bounds.lat.min()
    lat_max = 90
else:
    xr_out_bounds = xr_dataset.where(xr_dataset.lat <= LAT_BOUND_SOUTH,
                                     drop=True)
    lat_min = max(-79, xr_out_bounds.lat.min())
    lat_max = xr_out_bounds.lat.max()


if DEBUG:
    print("OUTSIDE BOUNDARY LAT")
    print(xr_out_bounds)
    print_min_max(xr_out_bounds)
    print('\n\n')

# create output grid using lat/lon bounds of data outside boundary
out_grid = xe.util.grid_2d(0,
                           360,
                           LONGITUDE_SPACING,
```

(continues on next page)

(continued from previous page)

```
                          lat_min,
                          lat_max,
                          LATITUDE_SPACING)

# create regridder using cropped data and output grid
# NOTE: this creates a temporary file in the current directory!
# consider supplying path to file in tmp directory using filename arg
# set reuse_weights=True to read temporary weight file if it exists
regridder = xe.Regridder(xr_out_bounds,
                         out_grid,
                         'bilinear',
                         ignore_degenerate=True,
                         reuse_weights=True,
                         filename=weight_file)

# regrid data
xr_out_regrid = regridder(xr_out_bounds)
met_data = xr_out_regrid[var]

# flip the data
met_data = met_data[::-1, ]

if DEBUG:
    print("PRINT MET DATA")
    print(met_data)

    print("Data Shape: " + repr(met_data.shape))
    print("Data Type:  " + repr(met_data.dtype))
    print("Max: " + repr(met_data.max))
    print_min_max(met_data)
    print('\n\n')

# Calculate attributes
lat_lower_left = float(met_data['lat'].min())
lon_lower_left = float(met_data['lon'].min())
n_lat = met_data['lat'].shape[0]
n_lon = met_data['lon'].shape[1]
delta_lat = (float(met_data['lat'].max()) - float(met_data['lat'].min()))/float(n_lat)
delta_lon = (float(met_data['lon'].max()) - float(met_data['lon'].min()))/float(n_lon)

# create the attributes dictionary to describe the data to pass to MET
met_data.attrs = {
        'valid': valid_time,
        'init': valid_time,
        'lead': "00",
```

(continues on next page)

(continued from previous page)

```
        'accum': "00",
        'name': var,
        'standard_name': standard_name,
        'long_name': long_name,
        'level': "SURFACE",
        'units': "UNKNOWN",

        # Definition for LatLon grid
        'grid': {
            'type': "LatLon",
            'name': "RTOFS Grid",
            'lat_ll': lat_lower_left,
            'lon_ll': lon_lower_left,
            'delta_lat': delta_lat,
            'delta_lon': delta_lon,
            'Nlat': n_lat,
            'Nlon': n_lon,
            }
        }
attrs = met_data.attrs
print("Attributes: " + repr(met_data.attrs))
```

Running METplus

This use case can be run two ways:

1) Passing in PlotDataPlane_obsHYCOM_coordTripolar.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/PlotDataPlane_obsHYCOM_coordTripolar.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PlotDataPlane_obsHYCOM_coordTripolar.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/PlotDataPlane_obsHYCOM_coordTripolar.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for thisIce use case will be found in model_applications/PlotDataPlane_obsHYCOM_coordTripolar (relative to **OUTPUT_BASE**) and will contain the following files:

- HYCOM_iceCoverage_north.ps
- HYCOM_iceCoverage_south.ps

Keywords

Note:

- PlotDataPlaneToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/marine_and_cryosphere-PlotDataPlane_obsHYCOM_coordTripolar.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.6.7 GridStat: Python Embedding to read and process ice cover

model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsOSTIA_iceCover.conf

Scientific Objective

This use case utilizes Python embedding to extract several statistics from the ice cover data over both pole regions, which was already being done in a closed system. By producing the same output via METplus, this use case provides standardization and reproducible results.

Datasets

Forecast: RTOFS ice cover file via Python Embedding script/file

Observation: OSTIA ice cover file via Python Embedding script/file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1165) section for more information.

Data Source: JPL's PODAAC and NCEP's FTPPRD data servers

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- scikit-learn
- pyproj
- pyresample

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case utilizes the METplus GridStat wrapper to generate a command to run the MET tool GridStat with Python Embedding for the specified user hemispheres

METplus Workflow

GridStat is the only tool called in this example. This use case will pass in the two hemispheres via a custom loop list, with both the observation and forecast gridded data being pulled from the files via Python Embedding. All of the desired statistics reside in the CNT line type, so that is the only output requested. It processes the following run time:

Valid: 2021-03-05 0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsOSTIA_iceCover.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsOSTIA_iceCover.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG=20210305
VALID_END=20210305
VALID_INCREMENT = 1M

LEAD_SEQ = 0

GRID_STAT_CUSTOM_LOOP_LIST = north, south


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

GRID_STAT_ONCE_PER_FIELD = False

MODEL = RTOFS

OBTYPE = UKMO

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsOSTIA_iceCover
```

(continues on next page)

(continued from previous page)

```
FCST_VAR1_NAME = {CONFIG_DIR}/read_ice_data.py {INPUT_BASE}/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsOSTIA_iceCover/{valid?fmt=%Y%m%d}_rtofs_glo_2ds_n024_ice.
↪nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsOSTIA_
↪iceCover/{valid?fmt=%Y%m%d}12_UKMO_L4.nc {custom} fcst

OBS_VAR1_NAME = {CONFIG_DIR}/read_ice_data.py {INPUT_BASE}/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsOSTIA_iceCover/{valid?fmt=%Y%m%d}_rtofs_glo_2ds_n024_ice.
↪nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsOSTIA_
↪iceCover/{valid?fmt=%Y%m%d}12_UKMO_L4.nc {custom} obs


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_REGRID_TO_GRID = NONE

GRID_STAT_GRID_WEIGHT_FLAG = AREA

GRID_STAT_DESC = NA

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_OUTPUT_PREFIX = {custom}

GRID_STAT_OUTPUT_FLAG_CNT = BOTH
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
```

(continues on next page)

(continued from previous page)

```
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
```

(continues on next page)

(continued from previous page)

```
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses one Python script to read forecast and observation data

parm/use_cases/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsOSTIA_iceCover/read_ice_data

```
#!/bin/env python
"""
Code adapted from
Todd Spindler
NOAA/NWS/NCEP/EMC

Designed to read in RTOFS and OSTIA data
and based on user input, process Arctic or Antarctic regions
for ice cover, and pass back in memory the forecast or observation
data field
"""

import numpy as np
from sklearn.metrics import mean_squared_error
import xarray as xr
import pandas as pd
from pyproj import Geod
import pyresample as pyr
from datetime import datetime, date
import os, sys

#------------------------------------
def iceArea(lon1,lat1,ice1):
    """
```

(continues on next page)

(continued from previous page)

```
    Compute the cell side dimensions (Vincenty) and the cell surface areas.
    This assumes the ice has already been masked and subsampled as needed
    returns ice_extent, ice_area, surface_area = iceArea(lon,lat,ice)
    surface_area is the computed grid areas in km**2)
    """
    lon=lon1.copy()
    lat=lat1.copy()
    ice=ice1.copy()
    g=Geod(ellps='WGS84')
    _,_,xdist=g.inv(lon,lat,np.roll(lon,-1,axis=1),np.roll(lat,-1,axis=1))
    _,_,ydist=g.inv(lon,lat,np.roll(lon,-1,axis=0),np.roll(lat,-1,axis=0))
    xdist=np.ma.array(xdist,mask=ice.mask)/1000.
    ydist=np.ma.array(ydist,mask=ice.mask)/1000.
    xdist=xdist[:-1,:-1]
    ydist=ydist[:-1,:-1]
    ice=ice[:-1,:-1]     # just to match the roll
    extent=xdist*ydist   # extent is surface area only
    area=xdist*ydist*ice # ice area is actual ice cover (area * concentration)
    return extent.flatten().sum(), area.flatten().sum(), extent

#----------------------------------------------------------

try:
    rtofsfile, icefile, hemisphere, file_flag = sys.argv[1:]
except ValueError:
    print("Must specify the following elements: fcst_file obs_file hemisphere file_flag")
    sys.exit(1)

HEMISPHERES = ['north', 'south']
FILE_FLAGS = ['fcst', 'obs']

if hemisphere not in HEMISPHERES or file_flag not in FILE_FLAGS:
    print(f"ERROR: Invalid hemisphere value ({hemisphere}) or file_flag value ({file_flag}) "
          f"Valid options are {HEMISPHERES} {FILE_FLAGS}")
    sys.exit(1)

print('processing',hemisphere+'ern hemisphere')
if hemisphere == 'north':
    bounding_lat=30.98
else:
    bounding_lat=-39.23


# load rtofs data and subset to hemisphere of interest and ice cover min value
print('reading rtofs ice')
```

(continues on next page)

(continued from previous page)

```
if not os.path.exists(rtofsfile):
    print('missing rtofs file',rtofsfile)
    sys.exit(1)
rtofs=xr.open_dataset(rtofsfile,decode_times=True)
rtofs=rtofs.ice_coverage[0,:-1,]

# load OSTIA data
print('reading OSTIA ice')
if not os.path.exists(icefile):
    print('missing OSTIA ice file',icefile)
    sys.exit(1)
ncep=xr.open_dataset(icefile,decode_times=True)
ncep=ncep.rename({'lon':'Longitude','lat':'Latitude'})
ncep=ncep.sea_ice_fraction.squeeze()

# trim to polar regions
if hemisphere == 'north':
    rtofs=rtofs.where((rtofs.Latitude>=bounding_lat),drop=True)
    ncep=ncep.where((ncep.Latitude>=bounding_lat),drop=True)
else:
    rtofs=rtofs.where((rtofs.Latitude<=bounding_lat),drop=True)
    ncep=ncep.where((ncep.Latitude<=bounding_lat),drop=True)

# now it's back to masked arrays for the RTOFS data
rlon=rtofs.Longitude.values
rlat=rtofs.Latitude.values
rice=rtofs.to_masked_array()

nlon=ncep.Longitude.values%360. # shift from -180 - 180 to 0-360
nlat=ncep.Latitude.values
nlon,nlat=np.meshgrid(nlon,nlat)  # shift from 1-d to 2-d arrays
nice=ncep.to_masked_array()

# mask out values below 15%
rice.mask=np.ma.mask_or(rice.mask,rice<0.15)
nice.mask=np.ma.mask_or(nice.mask,nice<0.15)

# compute ice area on original grids
print('computing ice area')
ncep_extent,ncep_area,ncep_surface_area=iceArea(nlon,nlat,nice)
rtofs_extent,rtofs_area,rtofs_surface_area=iceArea(rlon,rlat,rice)

# interpolate rtofs to ncep grid
print('interpolating rtofs to OSTIA grid')
```

(continues on next page)

(continued from previous page)

```
# pyresample gausssian-weighted kd-tree interp
rlon1=pyr.utils.wrap_longitudes(rlon)
rlat1=rlat.copy()
nlon1=pyr.utils.wrap_longitudes(nlon)
nlat1=nlat.copy()
# define the grids
orig_def = pyr.geometry.GridDefinition(lons=rlon1,lats=rlat1)
targ_def = pyr.geometry.GridDefinition(lons=nlon1,lats=nlat1)
radius=50000
sigmas=25000
rice2=pyr.kd_tree.resample_gauss(orig_def,rice,targ_def,
                                    radius_of_influence=radius,
                                    sigmas=sigmas,
                                    nprocs=8,
                                    neighbours=8,
                                    fill_value=None)

print('creating combined mask')
combined_mask=np.logical_and(nice.mask,rice2.mask)
nice2=nice.filled(fill_value=0.0)
rice2=rice2.filled(fill_value=0.0)
nice2=np.ma.array(nice2,mask=combined_mask)
rice2=np.ma.array(rice2,mask=combined_mask)

#Create the MET grids based on the file_flag
if file_flag == 'fcst':
    met_data = rice2[:-1,:-1]
    met_data = met_data[::-1,]
    #trim the lat/lon grids so they match the data fields
    #note that nice1 lat/lon fields are valid, since rice2 is interpolated to nice2
    lat_met = nlat1[:-1,:-1]
    lon_met = nlon1[:-1,:-1]
    print("Data shape: "+repr(met_data.shape))
    v_str = rtofsfile.split('_')[-6].split('/')[-1]
    v_str = v_str + '_120000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[1]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
```

(continues on next page)

(continued from previous page)

```
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'ice_coverage',
            'standard_name': rtofs.standard_name,
            'long_name': rtofs.long_name.strip(),
            'level': "SURFACE",
            'units': "UNKNOWN",

            'grid': {
                'type': "LatLon",
                'name': "RTOFS Grid",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs
if file_flag == 'obs':
    met_data = nice2[:-1,:-1]
    met_data = met_data[::-1,]
    #modify the lat and lon grids since they need to match the data dimensions, and code␣
↪cuts the last row/column of data
    lat_met = nlat1[:-1,:-1]
    lon_met = nlon1[:-1,:-1]
    print("Data shape: " +repr(met_data.shape))
    v_str = icefile.split('_')[-3].split('/')[-1]
    v_str = v_str[:-2]+'_120000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[1]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
```

(continues on next page)

(continued from previous page)

```
            'accum': "00",
            'name': 'ice_coverage',
            'standard_name': ncep.standard_name,
            'long_name': ncep.long_name.strip(),
            'level': "SURFACE",
            'units': "UNKNOWN",

            'grid': {
                'type': "LatLon",
                'name': "RTOFS Grid",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstRTOFS_obsOSTIA_iceCover.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsOSTIA_iceCover.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstRTOFS_obsOSTIA_iceCover.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsOSTIA_iceCover.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for thisIce use case will be found in 20210305 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_north_000000L_20210305_120000V_cnt.txt
- grid_stat_south_000000L_20210305_120000V_cnt.txt
- grid_stat_north_000000L_20210305_120000V.stat
- grid_stat_south_000000L_20210305_120000V.stat

Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/marine_and_cryosphere-GridStat_fcstRTOFS_obsOSTIA_iceCover.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.6.8 PointStat: read in buoy ASCII files to compare to model wave heights

model_applications/marine_and_cryosphere/PointStat_fcstGFS_obsNDBC_WaveHeight.conf

Scientific Objective

This use case utilizes the new ASCII2NC method to natively read in NDBC ASCII files, a common source of sea surface data for operational entities. These values are then compared to GFS' new wave height output, which it incorporated from Wave Watch III.

Datasets

Forecast: GFSv16 forecast data from WAVE file category

Observations: ASCII buoy files from NDBC

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1176) section for more information.

METplus Components

This use case calls ASCII2NC to read in ASCII buoy files and then PointStat for verification against GFS model data

METplus Workflow

ASCII2NC is the first tool called. It pulls in all files with a .txt type, which is the ASCII buoy data saved format. These observations are converted into a netCDF, which is then called by PointStat as the observation dataset. PointStat also pulls in a 3 hour forecast from the GFS for wave heights, which is included in the range of available buoy observation times. A +/- 30 minute window is allowed for the observational data. Thresholds are set that correspond to operational usage, and the CTC and CTS line types are requested. It processes the following run time:

Valid: 2022-10-16 09Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/marine_and_cryosphere/PointStat_fcstGFS_obsNDBC_WaveHeight.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/marine_and_
↪cryosphere/PointStat_fcstGFS_obsNDBC_WaveHeight.html
#
# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = ASCII2NC, PointStat

###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2022101609
VALID_END = 2022101609
VALID_INCREMENT = 1M

LEAD_SEQ = 0

LOOP_ORDER = times


###
# File I/O
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

ASCII2NC_INPUT_DIR = {INPUT_BASE}/model_applications/marine_and_cryosphere/PointStat_fcstGFS_
↪obsNDBC_WaveHeight
ASCII2NC_INPUT_TEMPLATE = *.txt

ASCII2NC_OUTPUT_DIR =
ASCII2NC_OUTPUT_TEMPLATE = {OUTPUT_BASE}/buoy_ASCII/buoy_{valid?fmt=%Y%m%d%H}.nc

ASCII2NC_SKIP_IF_OUTPUT_EXISTS = False

ASCII2NC_FILE_WINDOW_BEGIN = 0
ASCII2NC_FILE_WINDOW_END = 0

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/marine_and_cryosphere/PointStat_
↪fcstGFS_obsNDBC_WaveHeight
FCST_POINT_STAT_INPUT_TEMPLATE = gfswave.t06z.global.0p16.f003.grib2

OBS_POINT_STAT_INPUT_DIR =
OBS_POINT_STAT_INPUT_TEMPLATE = {ASCII2NC_OUTPUT_TEMPLATE}

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/PointStat


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

POINT_STAT_ONCE_PER_FIELD = False


FCST_VAR1_NAME = WVHGT
FCST_VAR1_LEVELS = Z0
BOTH_VAR1_THRESH = le3.0,ge4.0&&le6.0,ge8.0
OBS_VAR1_NAME = WVHT
OBS_VAR1_LEVELS = L0


###
# ASCII2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ascii2nc
###
```

(continues on next page)

(continued from previous page)

```
#LOG_ASCII2NC_VERBOSITY = 1

ASCII2NC_CONFIG_FILE = {PARM_BASE}/met_config/Ascii2NcConfig_wrapped

ASCII2NC_INPUT_FORMAT = ndbc_standard

ASCII2NC_MASK_GRID =
ASCII2NC_MASK_POLY =
ASCII2NC_MASK_SID =

ASCII2NC_TIME_SUMMARY_FLAG = False
ASCII2NC_TIME_SUMMARY_RAW_DATA = False
ASCII2NC_TIME_SUMMARY_BEG = 000000
ASCII2NC_TIME_SUMMARY_END = 235959
ASCII2NC_TIME_SUMMARY_STEP = 300
ASCII2NC_TIME_SUMMARY_WIDTH = 600
ASCII2NC_TIME_SUMMARY_GRIB_CODES = 11, 204, 211
ASCII2NC_TIME_SUMMARY_VAR_NAMES =
ASCII2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80
ASCII2NC_TIME_SUMMARY_VALID_FREQ = 0
ASCII2NC_TIME_SUMMARY_VALID_THRESH = 0.0

###
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###

#LOG_POINT_STAT_VERBOSITY = 2

POINT_STAT_CONFIG_FILE ={PARM_BASE}/met_config/PointStatConfig_wrapped


#POINT_STAT_OUTPUT_FLAG_FHO =
POINT_STAT_OUTPUT_FLAG_CTC = BOTH
POINT_STAT_OUTPUT_FLAG_CTS = BOTH
OBS_POINT_STAT_WINDOW_BEGIN = -1800
OBS_POINT_STAT_WINDOW_END = 1800

POINT_STAT_OFFSETS = 0

MODEL = GFSv16

POINT_STAT_DESC = NDBC
OBTYPE =
```

(continues on next page)

(continued from previous page)

```
POINT_STAT_REGRID_TO_GRID = NONE
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2

POINT_STAT_MESSAGE_TYPE = NDBC_STANDARD

POINT_STAT_MASK_GRID = FULL
POINT_STAT_MASK_POLY =
POINT_STAT_MASK_SID =

[user_env_vars]
MET_NDBC_STATIONS = {INPUT_BASE}/model_applications/marine_and_cryosphere/PointStat_fcstGFS_
↪obsNDBC_WaveHeight/ndbc_stations.20220928.xml
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default ascii2nc configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to summarize the ASCII data read in
//

//
// Time periods for the summarization
// obs_var (string array) is added and works like grib_code (int array)
// when the obs name is given instead of grib_code
//
```

(continues on next page)

(continued from previous page)

```
${METPLUS_TIME_SUMMARY_DICT}

//
// Mapping of input little_r report types to output message types
//
message_type_map = [
   { key = "FM-12 SYNOP";  val = "ADPSFC"; },
   { key = "FM-13 SHIP";   val = "SFCSHP"; },
   { key = "FM-15 METAR";  val = "ADPSFC"; },
   { key = "FM-18 BUOY";   val = "SFCSHP"; },
   { key = "FM-281 QSCAT"; val = "ASCATW"; },
   { key = "FM-32 PILOT";  val = "ADPUPA"; },
   { key = "FM-35 TEMP";   val = "ADPUPA"; },
   { key = "FM-88 SATOB";  val = "SATWND"; },
   { key = "FM-97 ACARS";  val = "AIRCFT"; }
];

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
```

(continues on next page)

(continued from previous page)

```
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
```

(continues on next page)

(continued from previous page)

```
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
```

(continues on next page)

(continued from previous page)

```
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in PointStat_fcstGFS_obsNDBC_WaveHeight.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/PointStat_fcstGFS_obsNDBC_WaveHeight.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PointStat_fcstGFS_obsNDBC_WaveHeight.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/PointStat_fcstGFS_obsNDBC_WaveHeight.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in PointStat and buoy_ASCII directories (relative to **OUTPUT_BASE**) and will contain the following files:

- point_stat_030000L_20221016_090000V_ctc.txt
- point_stat_030000L_20221016_090000V_cts.txt
- point_stat_030000L_20221016_090000V.stat
- buoy_2022101609.nc

Keywords

Note:

- PointStatToolUseCase
- ASCII2NCToolUseCase
- GRIB2FileUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/marine_and_cryosphere-PointStat_fcstGFS_obsNDBC_WaveHeight.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.6.9 PointStat: Python embedding to read Argo netCDF files to verify ocean temperature forecast at 50 m depth

model_applications/marine_and_cryosphere/PointStat_fcstRTOFS_obsARGO_climoWOA23_temp.conf

Scientific Objective

This use case utilizes the ASCII2NC tool with python embedding to natively read in Argo netCDF files, a common source of ocean profile data for operational entities. These values are then used by the PointStat tool to verify RTOFS ocean temperature forecast at 50 m depth.

Datasets

Forecast: RTOFSv2.3 forecast data pre-processed into 0.1 degree lat-lon grid

Observations: three netCDF files from Argo

Climatology: two monthly climatology files from WOA23

Sea Ice Mask: a mask file to exclude forecast grid points with sea ice concentration > 15%

Location: All of the input data required for this use case can be found in the marine_and_cryosphere sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1193) section for more information.

METplus Components

This use case calls ASCII2NC to read in Argo netCDF files and then PointStat for verification against RTOFS model data.

METplus Workflow

ASCII2NC is the first tool called. It pulls in three Argo files for the Atlantic, Pacific, and Indian Oceans, respectively using a Python script. These observations are converted into a netCDF file, which is then called by PointStat as the observation dataset. PointStat also pulls in a forecast from the RTOFS for ocean temperature at 50 m depth, which is included in the range of available observation times, and two monthly climatology files from the WOA23 to calculate daily climatology for the valid date. A 24-hour (18Z to 18Z) window is allowed for the observational data, and a "UNIQUE" flag is set to only use the observational

data closest to the forecast valid time at a given location. Temperature thresholds are set to correspond to operational usage, and the CTC, CTS, CNT, SL1L2, and SAL1L2 line types are requested. It processes the following run time:

Valid: 2023-03-18 00Z

Forecast lead: 24 hour

METplus Configuration

METplus first loads all of the default configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/marine_and_cryosphere/PointStat_fcstRTOFS_obsARGO_climoWOA23_temp.co

```
###
# Purpose: Example METplus configuration file to verify RTOFS ocean temperature
#          forecasts at 50 m depth with Argo profile data and WOA23 climatology
#          using python embedding.
# Contributors: L. Gwen Chen (lichuan.chen@noaa.gov), George McCabe,
#               John Halley Gotway, and Daniel Adriaansen
# Date: 22 March 2023
###

[config]

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = ASCII2NC,PointStat

###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###
```

(continues on next page)

(continued from previous page)

```
LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20230318
VALID_END = 20230318
VALID_INCREMENT = 24H

LEAD_SEQ = 024

###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

PY_EMBED_SCRIPT = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/PointStat_
↪fcstRTOFS_obsARGO_climoWOA23_temp/read_argo_metplus.py

INPUT_FILE = {INPUT_BASE}/model_applications/marine_and_cryosphere/PointStat_fcstRTOFS_
↪obsARGO_climoWOA23_temp/argo/atlantic_ocean/{valid?fmt=%Y%m%d}_prof.nc {INPUT_BASE}/model_
↪applications/marine_and_cryosphere/PointStat_fcstRTOFS_obsARGO_climoWOA23_temp/argo/indian_
↪ocean/{valid?fmt=%Y%m%d}_prof.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪PointStat_fcstRTOFS_obsARGO_climoWOA23_temp/argo/pacific_ocean/{valid?fmt=%Y%m%d}_prof.nc

ASCII2NC_INPUT_DIR =
ASCII2NC_INPUT_TEMPLATE = "{PY_EMBED_SCRIPT} {INPUT_FILE}"

ASCII2NC_OUTPUT_DIR = {OUTPUT_BASE}/prep
ASCII2NC_OUTPUT_TEMPLATE = argo.{valid?fmt=%Y%m%d}.nc

ASCII2NC_SKIP_IF_OUTPUT_EXISTS = False

ASCII2NC_FILE_WINDOW_BEGIN = 0
ASCII2NC_FILE_WINDOW_END = 0

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/marine_and_cryosphere/PointStat_
↪fcstRTOFS_obsARGO_climoWOA23_temp
FCST_POINT_STAT_INPUT_TEMPLATE = rtofs.{init?fmt=%Y%m%d}/rtofs_glo_3dz_f{lead?fmt=%3H}_daily_
↪3ztio.argo.nc

OBS_POINT_STAT_INPUT_DIR = {ASCII2NC_OUTPUT_DIR}
OBS_POINT_STAT_INPUT_TEMPLATE = {ASCII2NC_OUTPUT_TEMPLATE}

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/stats
POINT_STAT_OUTPUT_TEMPLATE = rtofs.{valid?fmt=%Y%m%d}
```

(continues on next page)

(continued from previous page)

```
###
# ASCII2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ascii2nc
###

#ASCII2NC_CONFIG_FILE =

ASCII2NC_WINDOW_BEGIN = 0
ASCII2NC_WINDOW_END = 0

ASCII2NC_INPUT_FORMAT = python

ASCII2NC_MASK_GRID =
ASCII2NC_MASK_POLY =
ASCII2NC_MASK_SID =

ASCII2NC_TIME_SUMMARY_FLAG = False
ASCII2NC_TIME_SUMMARY_RAW_DATA = False
ASCII2NC_TIME_SUMMARY_BEG = 000000
ASCII2NC_TIME_SUMMARY_END = 235959
ASCII2NC_TIME_SUMMARY_STEP = 300
ASCII2NC_TIME_SUMMARY_WIDTH = 600
ASCII2NC_TIME_SUMMARY_GRIB_CODES = 11, 204, 211
ASCII2NC_TIME_SUMMARY_VAR_NAMES =
ASCII2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80
ASCII2NC_TIME_SUMMARY_VALID_FREQ = 0
ASCII2NC_TIME_SUMMARY_VALID_THRESH = 0.0

###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

POINT_STAT_ONCE_PER_FIELD = False

FCST_VAR1_NAME = temperature
FCST_VAR1_LEVELS = "(0,@50,*,*)"
FCST_VAR1_OPTIONS = set_attr_lead = "{lead?fmt=%3H}"; set_attr_level = "Z50";
OBS_VAR1_NAME = TEMP
OBS_VAR1_LEVELS = Z48-52
OBS_VAR1_OPTIONS = set_attr_units = "degC";
BOTH_VAR1_THRESH = >=0, >=26.5

###
# PointStat Settings
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###

OBS_POINT_STAT_WINDOW_BEGIN = -21600
OBS_POINT_STAT_WINDOW_END = 64800
POINT_STAT_MET_CONFIG_OVERRIDES = duplicate_flag = UNIQUE; obs_summary = NEAREST;

POINT_STAT_OFFSETS = 0

MODEL = RTOFS
OBTYPE = ARGO
POINT_STAT_DESC = NA
POINT_STAT_OUTPUT_PREFIX = {MODEL}_{OBTYPE}_temp_Z50

POINT_STAT_REGRID_TO_GRID = NONE
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2

POINT_STAT_MESSAGE_TYPE = ARGO

POINT_STAT_MASK_GRID =
POINT_STAT_MASK_SID =
POINT_STAT_MASK_POLY = {INPUT_BASE}/model_applications/marine_and_cryosphere/PointStat_
↪fcstRTOFS_obsARGO_climoWOA23_temp/rtofs.{init?fmt=%Y%m%d}/mask.global.nc

# Set up climatology files and interpolation methods
POINT_STAT_CLIMO_MEAN_FILE_NAME = {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪PointStat_fcstRTOFS_obsARGO_climoWOA23_temp/woa23/woa23_decav91C0_t03_04.nc, {INPUT_BASE}/
↪model_applications/marine_and_cryosphere/PointStat_fcstRTOFS_obsARGO_climoWOA23_temp/woa23/
↪woa23_decav91C0_t04_04.nc
POINT_STAT_CLIMO_MEAN_FIELD = {name = "t_an"; level = "(0,@50,*,*)";}
POINT_STAT_CLIMO_MEAN_REGRID_METHOD = BILIN
POINT_STAT_CLIMO_MEAN_REGRID_WIDTH = 2
POINT_STAT_CLIMO_MEAN_REGRID_VLD_THRESH = 0.5
POINT_STAT_CLIMO_MEAN_REGRID_SHAPE = SQUARE
POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = DW_MEAN
POINT_STAT_CLIMO_MEAN_DAY_INTERVAL = 31
POINT_STAT_CLIMO_MEAN_HOUR_INTERVAL = 6

POINT_STAT_CLIMO_CDF_WRITE_BINS = False

# Set up output files
POINT_STAT_OUTPUT_FLAG_CTC = STAT
POINT_STAT_OUTPUT_FLAG_CTS = STAT
POINT_STAT_OUTPUT_FLAG_CNT = STAT
```

(continues on next page)

(continued from previous page)

```
POINT_STAT_OUTPUT_FLAG_SL1L2 = STAT
POINT_STAT_OUTPUT_FLAG_SAL1L2 = STAT
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *PointStat MET Configuration* (page 218) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default ascii2nc configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to summarize the ASCII data read in
//

//
// Time periods for the summarization
// obs_var (string array) is added and works like grib_code (int array)
// when the obs name is given instead of grib_code
//
${METPLUS_TIME_SUMMARY_DICT}

//
// Mapping of input little_r report types to output message types
//
message_type_map = [
   { key = "FM-12 SYNOP";  val = "ADPSFC"; },
   { key = "FM-13 SHIP";   val = "SFCSHP"; },
   { key = "FM-15 METAR";  val = "ADPSFC"; },
   { key = "FM-18 BUOY";   val = "SFCSHP"; },
   { key = "FM-281 QSCAT"; val = "ASCATW"; },
   { key = "FM-32 PILOT";  val = "ADPUPA"; },
```

(continues on next page)

(continued from previous page)

```
   { key = "FM-35 TEMP";   val = "ADPUPA"; },
   { key = "FM-88 SATOB";  val = "SATWND"; },
   { key = "FM-97 ACARS";  val = "AIRCFT"; }
];

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
```

(continues on next page)

(continued from previous page)

```
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses one Python script to read observation data.

parm/use_cases/model_applications/marine_and_cryosphere/PointStat_fcstRTOFS_obsARGO_climoWOA23_temp/re

```
#! /usr/bin/env python3

import sys
import os
import datetime

import netCDF4
import numpy
from numpy.ma import is_masked

# set to true to output more info
DEBUG = False

# constant values that will be used for every observation
MESSAGE_TYPE = 'ARGO'
ELEVATION = 'NA'
LEVEL = 'NA'
QC_STRING = '1'

# skip data if PRES_ADJUSTED_ERROR value is greater than this value
MAX_PRESSURE_ERROR = 20

"""Read and format the input 11-column observations:
(1)  string:  Message_Type
(2)  string:  Station_ID
(3)  string:  Valid_Time(YYYYMMDD_HHMMSS)
(4)  numeric: Lat(Deg North)
(5)  numeric: Lon(Deg East)
(6)  numeric: Elevation(msl)
```

(continues on next page)

(continued from previous page)

```
(7)  string:  Var_Name(or GRIB_Code)
(8)  numeric: Level
(9)  numeric: Height(msl or agl)
(10) string:  QC_String
(11) numeric: Observation_Value
"""


def get_string_value(var):
    """!Get string value from NetCDF variable. The string variables are stored
    as bytes, so decode them and strip off any whitespace before or after.

    @param var NetCDF variable to read
    @returns string value from variable
    """
    return var[:].tobytes().decode('utf-8').strip()


def get_val_check_qc(nc_obj, field_name, idx_p, idx_l=None, error_max=None):
    """!Get field value unless quality control checks do not pass.
    The conditions to skip data are as follows:
    1) If {field_name}_QC is masked.
    2) If {field_name} is masked.
    3) If {field_name}_QC value is not equal to 1 (recommended from ARGO docs).
    4) If error_max is set, skip if {field_name}_ERROR is not masked and less
    than the error_max value.

    @param nc_obj NetCDF object
    @param field_name name of field to read
    @param idx_p index of profile (1st dimension of data)
    @param idx_l (optional) index of level (2nd dimension of data). Defaults to
    None which assumes field is not 2D
    @param error_max (optional) value to compare to {field_name}_ERROR. Skip if
    error value is less than this value
    @returns numpy.float64 field value if all QC checks pass or None
    """
    if idx_l is None:
        qc = nc_obj.variables[f'{field_name}_QC'][idx_p]
        field = nc_obj.variables[field_name][idx_p]
        data_str = f'{field_name}[{idx_p}]'
    else:
        qc = nc_obj.variables[f'{field_name}_QC'][idx_p][idx_l]
        field = nc_obj.variables[field_name][idx_p][idx_l]
        data_str = f'{field_name}[{idx_p}][{idx_l}]'
```

(continues on next page)

(continued from previous page)

```
    qc_mask = is_masked(qc)
    field_mask = is_masked(field)

    if qc_mask:
        if DEBUG:
            print(f"Skip {data_str} {field_name}_QC is masked")
        return None

    if field_mask:
        if DEBUG:
            print(f"Skip {data_str} {field_name} is masked")
        return None

    if int(qc) != 1:
        if DEBUG:
            print(f"Skip {data_str} {field_name}_QC value ({int(qc)}) != 1")
        return None

    if error_max:
        err = nc_obj.variables.get(f'{field_name}_ERROR')
        if err:
            err = err[idx_p] if idx_l is None else err[idx_p][idx_l]
            if not is_masked(err) and err > error_max:
                print(f"Skip {data_str} {field_name}_ERROR > {error_max}")
                return None

    return numpy.float64(field)


def get_valid_time(ref_dt, nc_obj, idx):
    """!Get valid time by adding julian days to the reference date time.

    @param ref_dt Datetime object of reference date time
    @param nc_obj NetCDF object
    @param idx index of profile to read
    @returns string of valid time in YYYYMMDD_HHMMSS format
    """
    julian_days = nc_obj.variables['JULD'][idx]
    day_offset = datetime.timedelta(days=float(julian_days))
    valid_dt = ref_dt + day_offset
    return valid_dt.strftime('%Y%m%d_%H%M%S')


def get_lat_lon(nc_obj, idx):
    """!Read latitude and longitude values from NetCDF file at profile index.
```

(continues on next page)

(continued from previous page)

```
    @param nc_obj NetCDF object
    @param idx index of profile to read
    @returns tuple of lat and lon values as floats
    """
    return (numpy.float64(nc_obj.variables['LATITUDE'][idx]),
            numpy.float64(nc_obj.variables['LONGITUDE'][idx]))


if len(sys.argv) < 2:
    print(f"ERROR: {__file__} - Must provide at least 1 input file argument")
    sys.exit(1)

is_ok = True
input_files = []
for arg in sys.argv[1:]:
    if arg.endswith('debug'):
        print('Debugging output turned on')
        DEBUG = True
        continue

    input_file = os.path.expandvars(arg)
    if not os.path.exists(input_file):
        print(f'ERROR: Input file does not exist: {input_file}')
        is_ok = False
        continue

    input_files.append(input_file)

if not is_ok:
    sys.exit(1)

print(f'Number of input files: {len(input_files)}')

point_data = []
for input_file in input_files:
    print(f'Processing file: {input_file}')

    nc_in = netCDF4.Dataset(input_file, 'r')

    # get reference date time
    time_str = get_string_value(nc_in.variables['REFERENCE_DATE_TIME'])
    reference_date_time = datetime.datetime.strptime(time_str, '%Y%m%d%H%M%S')

    # get number of profiles and levels
```

(continues on next page)

(continued from previous page)

```
    num_profiles = nc_in.dimensions['N_PROF'].size
    num_levels = nc_in.dimensions['N_LEVELS'].size

    new_point_data = []
    for index_p in range(0, num_profiles):
        # check QC and mask of JULD to skip profiles with bad time info
        if get_val_check_qc(nc_in, 'JULD', index_p) is None:
            continue

        valid_time = get_valid_time(reference_date_time, nc_in, index_p)
        station_id = get_string_value(
            nc_in.variables['PLATFORM_NUMBER'][index_p]
        )
        lat, lon = get_lat_lon(nc_in, index_p)

        # loop through levels
        for index_l in range(0, num_levels):
            # read pressure data to get height in meters of sea water (msw)
            height = get_val_check_qc(nc_in, 'PRES_ADJUSTED', index_p, index_l,
                                      error_max=MAX_PRESSURE_ERROR)
            if height is None:
                continue

            # get temperature and ocean salinity values
            for var_name in ('TEMP', 'PSAL'):
                observation_value = get_val_check_qc(nc_in,
                                                     f'{var_name}_ADJUSTED',
                                                     index_p, index_l)
                if observation_value is None:
                    continue

                point = [
                    MESSAGE_TYPE, station_id, valid_time, lat, lon, ELEVATION,
                    var_name, LEVEL, height, QC_STRING, observation_value,
                ]
                new_point_data.append(point)
                if DEBUG:
                    print(', '.join([str(val) for val in point]))

    point_data.extend(new_point_data)
    nc_in.close()

print("     point_data: Data Length:\t" + repr(len(point_data)))
print("     point_data: Data Type:\t" + repr(type(point_data)))
```

Running METplus

Pass the use case configuration file to the run_metplus.py script along with any user-specific system configuration files if desired:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/marine_and_cryosphere/
↪PointStat_fcstRTOFS_obsARGO_climoWOA23_temp.conf /path/to/user_system.conf
```

See *Running METplus* (page 26) for more information.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in prep and stats/rtofs.20230318 directories (relative to **OUTPUT_BASE**) and will contain the following files:

- argo.20230318.nc
- point_stat_RTOFS_ARGO_temp_Z50_240000L_20230318_000000V.stat

Keywords

Note:

- PointStatToolUseCase
- ASCII2NCToolUseCase
- PythonEmbeddingFileUseCase
- ClimatologyUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/marine_and_cryosphere-PointStat_fcstRTOFS_obsARGO_climoWOA23_tem

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.6.10 GridStat: Python Embedding for sea surface salinity using level 3, 8 day mean obs

model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss.conf

Scientific Objective

This use case utilizes Python embedding to extract several statistics from the sea surface salinity data over the globe, which was already being done in a closed system. By producing the same output via METplus, this use case provides standardization and reproducible results.

Datasets

Forecast: RTOFS sss file via Python Embedding script/file

Observations: SMAP sss file via Python Embedding script/file

Sea Ice Masking: RTOFS ice cover file via Python Embedding script/file

Climatology: WOA sss file via Python Embedding script/file

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1210) section for more information.

Data Source: JPL's PODAAC and NCEP's FTPPRD data servers

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- scikit-learn
- pyresample

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case utilizes the METplus GridStat wrapper to generate a command to run the MET tool GridStat with Python Embedding for the specified user hemispheres

METplus Workflow

GridStat is the only tool called in this example. This use case will pass in both the observation, forecast, and climatology gridded data being pulled from the files via Python Embedding. All of the desired statistics reside in the CNT line type, so that is the only output requested. It processes the following run time:

Valid: 2021-05-02 0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat
```

(continues on next page)

(continued from previous page)

```
###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG=20210502
VALID_END=20210502
VALID_INCREMENT = 1M

LEAD_SEQ = 24


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

GRID_STAT_ONCE_PER_FIELD = False

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/GridStat_
```

(continues on next page)

(continued from previous page)

```
↪fcstRTOFS_obsSMAP_climWOA_sss

FCST_VAR1_NAME = {CONFIG_DIR}/read_rtofs_smap_woa.py {INPUT_BASE}/model_applications/marine_
↪and_cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss/{init?fmt=%Y%m%d}_rtofs_glo_2ds_f024_
↪prog.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMAP_
↪climWOA_sss/SMAP-L3-GLOB_{valid?fmt=%Y%m%d?shift=86400}.nc {INPUT_BASE}/model_applications/
↪marine_and_cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss/OSTIA-UKMO-L4-GLOB-v2.0_
↪{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsSMAP_climWOA_sss {valid?fmt=%Y%m%d} fcst

OBS_VAR1_NAME = {CONFIG_DIR}/read_rtofs_smap_woa.py {INPUT_BASE}/model_applications/marine_
↪and_cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss/{init?fmt=%Y%m%d}_rtofs_glo_2ds_f024_
↪prog.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMAP_
↪climWOA_sss/SMAP-L3-GLOB_{valid?fmt=%Y%m%d?shift=86400}.nc {INPUT_BASE}/model_applications/
↪marine_and_cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss/OSTIA-UKMO-L4-GLOB-v2.0_
↪{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsSMAP_climWOA_sss {valid?fmt=%Y%m%d} obs

GRID_STAT_CLIMO_MEAN_FIELD = {name="{CONFIG_DIR}/read_rtofs_smap_woa.py {INPUT_BASE}/model_
↪applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss/{init?fmt=%Y%m%d}
↪_rtofs_glo_2ds_f024_prog.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsSMAP_climWOA_sss/SMAP-L3-GLOB_{valid?fmt=%Y%m%d?shift=86400}.nc {INPUT_BASE}/
↪model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss/OSTIA-UKMO-
↪L4-GLOB-v2.0_{valid?fmt=%Y%m%d}.nc {INPUT_BASE}/model_applications/marine_and_cryosphere/
↪GridStat_fcstRTOFS_obsSMAP_climWOA_sss {valid?fmt=%Y%m%d} climo"; level="(*,*)";}


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_REGRID_TO_GRID = NONE

MODEL = RTOFS
OBTYPE = SMAP

GRID_STAT_DESC = NA

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_OUTPUT_PREFIX = SSS

GRID_STAT_OUTPUT_FLAG_CNT = BOTH
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
```

(continues on next page)

(continued from previous page)

```
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
```

(continues on next page)

(continued from previous page)

```
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses one Python script to read forecast and observation data

parm/use_cases/model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss/read_rtof

```
#!/bin/env python
"""
Code adapted from
Todd Spindler
NOAA/NWS/NCEP/EMC
Designed to read in RTOFS,SMAP,WOA and OSTIA data
and based on user input, read sss data
and pass back in memory the forecast, observation, or climatology
data field
"""
```

(continues on next page)

(continued from previous page)

```
import numpy as np
import xarray as xr
import pandas as pd
import pyresample as pyr
from pandas.tseries.offsets import DateOffset
from datetime import datetime, timedelta
from sklearn.metrics import mean_squared_error
import io
from glob import glob
import warnings
import os, sys


if len(sys.argv) < 6:
    print("Must specify the following elements: fcst_file obs_file ice_file, climo_file,␣
↪valid_date, file_flag")
    sys.exit(1)

rtofsfile = os.path.expandvars(sys.argv[1])
sssfile = os.path.expandvars(sys.argv[2])
icefile = os.path.expandvars(sys.argv[3])
climoDir = os.path.expandvars(sys.argv[4])
vDate=datetime.strptime(sys.argv[5],'%Y%m%d')
file_flag = sys.argv[6]

print('Starting Satellite SMAP V&V at',datetime.now(),'for',vDate, ' file_flag:',file_flag)

pd.date_range(vDate,vDate)
platform='SMAP'
param='sss'


######################################################################
# READ SMAP data #####################################################
######################################################################

if not os.path.exists(sssfile):
        print('missing SMAP file for',vDate)

sss_data=xr.open_dataset(sssfile,decode_times=True)
sss_data['time']=sss_data.time-pd.Timedelta('12H')  # shift 12Z offset time to 00Z
sss_data2=sss_data['sss'].astype('single')
print('Retrieved SMAP data from NESDIS for',sss_data2.time.values)
#sss_data2=sss_data2.rename({'longitude':'lon','latitude':'lat'})
```

(continues on next page)

(continued from previous page)

```
# all coords need to be single precision
sss_data2['lon']=sss_data2.lon.astype('single')
sss_data2['lat']=sss_data2.lat.astype('single')
sss_data2.attrs['platform']=platform
sss_data2.attrs['units']='PSU'

######################################################################
# READ RTOFS data (model output in Tri-polar coordinates) ############
######################################################################

print('reading rtofs ice')
if not os.path.exists(rtofsfile):
    print('missing rtofs file',rtofsfile)
    sys.exit(1)

indata=xr.open_dataset(rtofsfile,decode_times=True)


indata=indata.mean(dim='MT')
indata = indata[param][:-1,]
indata.coords['time']=vDate
#indata.coords['fcst']=fcst

outdata=indata.copy()

outdata=outdata.rename({'Longitude':'lon','Latitude':'lat',})
# all coords need to be single precision
outdata['lon']=outdata.lon.astype('single')
outdata['lat']=outdata.lat.astype('single')
outdata.attrs['platform']='rtofs '+platform

######################################################################
# READ CLIMO WOA data - May require 2 files depending on the date ###
######################################################################

if not os.path.exists(climoDir):
        print('missing climo file file for',vDate)

vDate=pd.Timestamp(vDate)

climofile="woa13_decav_s{:02n}_04v2.nc".format(vDate.month)
climo_data=xr.open_dataset(climoDir+'/'+climofile,decode_times=False)
climo_data=climo_data['s_an'].squeeze()[0,]
```

(continues on next page)

(continued from previous page)

```
if vDate.day==15:  # even for Feb, just because
    climofile="woa13_decav_s{:02n}_04v2.nc".format(vDate.month)
    climo_data=xr.open_dataset(climoDir+'/'+climofile,decode_times=False)
    climo_data=climo_data['s_an'].squeeze()[0,]  # surface only
else:
    if vDate.day < 15:
        start=vDate - DateOffset(months=1,day=15)
        stop=pd.Timestamp(vDate.year,vDate.month,15)
    else:
        start=pd.Timestamp(vDate.year,vDate.month,15)
        stop=vDate + DateOffset(months=1,day=15)
    left=(vDate-start)/(stop-start)

    climofile1="woa13_decav_s{:02n}_04v2.nc".format(start.month)
    climofile2="woa13_decav_s{:02n}_04v2.nc".format(stop.month)
    climo_data1=xr.open_dataset(climoDir+'/'+climofile1,decode_times=False)
    climo_data2=xr.open_dataset(climoDir+'/'+climofile2,decode_times=False)
    climo_data1=climo_data1['s_an'].squeeze()[0,]  # surface only
    climo_data2=climo_data2['s_an'].squeeze()[0,]  # surface only

    print('climofile1 :', climofile1)
    print('climofile2 :', climofile2)
    climo_data=climo_data1+((climo_data2-climo_data1)*left)
    climofile='weighted average of '+climofile1+' and '+climofile2

# all coords need to be single precision
climo_data['lon']=climo_data.lon.astype('single')
climo_data['lat']=climo_data.lat.astype('single')
climo_data.attrs['platform']='woa'
climo_data.attrs['filename']=climofile

######################################################################
# READ ICE data for masking ##########################################
######################################################################

if not os.path.exists(icefile):
        print('missing OSTIA ice file for',vDate)

ice_data=xr.open_dataset(icefile,decode_times=True)
ice_data=ice_data.rename({'sea_ice_fraction':'ice'})

# all coords need to be single precision
ice_data2=ice_data.ice.astype('single')
ice_data2['lon']=ice_data2.lon.astype('single')
```

(continues on next page)

(continued from previous page)

```
ice_data2['lat']=ice_data2.lat.astype('single')


def regrid(model,obs):
    """
    regrid data to obs -- this assumes DataArrays
    """
    model2=model.copy()
    model2_lon=model2.lon.values
    model2_lat=model2.lat.values
    model2_data=model2.to_masked_array()
    if model2_lon.ndim==1:
        model2_lon,model2_lat=np.meshgrid(model2_lon,model2_lat)

    obs2=obs.copy()
    obs2_lon=obs2.lon.astype('single').values
    obs2_lat=obs2.lat.astype('single').values
    obs2_data=obs2.astype('single').to_masked_array()
    if obs2.lon.ndim==1:
        obs2_lon,obs2_lat=np.meshgrid(obs2.lon.values,obs2.lat.values)

    model2_lon1=pyr.utils.wrap_longitudes(model2_lon)
    model2_lat1=model2_lat.copy()
    obs2_lon1=pyr.utils.wrap_longitudes(obs2_lon)
    obs2_lat1=obs2_lat.copy()

    # pyresample gausssian-weighted kd-tree interp
    # define the grids
    orig_def = pyr.geometry.GridDefinition(lons=model2_lon1,lats=model2_lat1)
    targ_def = pyr.geometry.GridDefinition(lons=obs2_lon1,lats=obs2_lat1)
    radius=50000
    sigmas=25000
    model2_data2=pyr.kd_tree.resample_gauss(orig_def,model2_data,targ_def,
                                            radius_of_influence=radius,
                                            sigmas=sigmas,
                                            fill_value=None)
    model=xr.DataArray(model2_data2,coords=[obs.lat.values,obs.lon.values],dims=['lat','lon
↪'])

    return model

def expand_grid(data):
    """
    concatenate global data for edge wraps
    """
```

(continues on next page)

(continued from previous page)

```
    data2=data.copy()
    data2['lon']=data2.lon+360
    data3=xr.concat((data,data2),dim='lon')
    return data3

sss_data2=sss_data2.squeeze()

print('regridding climo to obs')
climo_data=climo_data.squeeze()
climo_data=regrid(climo_data,sss_data2)

print('regridding ice to obs')
ice_data2=regrid(ice_data2,sss_data2)

print('regridding model to obs')
model2=regrid(outdata,sss_data2)

# combine obs ice mask with ncep
obs2=sss_data2.to_masked_array()
ice2=ice_data2.to_masked_array()
climo2=climo_data.to_masked_array()
model2=model2.to_masked_array()

#reconcile with obs
obs2.mask=np.ma.mask_or(obs2.mask,ice2>0.0)
obs2.mask=np.ma.mask_or(obs2.mask,climo2.mask)
obs2.mask=np.ma.mask_or(obs2.mask,model2.mask)
climo2.mask=obs2.mask
model2.mask=obs2.mask

obs2=xr.DataArray(obs2,coords=[sss_data2.lat.values,sss_data2.lon.values], dims=['lat','lon
↪'])
model2=xr.DataArray(model2,coords=[sss_data2.lat.values,sss_data2.lon.values], dims=['lat',
↪'lon'])
climo2=xr.DataArray(climo2,coords=[sss_data2.lat.values,sss_data2.lon.values], dims=['lat',
↪'lon'])

model2=expand_grid(model2)
climo2=expand_grid(climo2)
obs2=expand_grid(obs2)

#Create the MET grids based on the file_flag
if file_flag == 'fcst':
    met_data = model2[:,:]
```

(continues on next page)

(continued from previous page)

```
    #trim the lat/lon grids so they match the data fields
    lat_met = model2.lat
    lon_met = model2.lon
    print(" RTOFS Data shape: "+repr(met_data.shape))
    v_str = vDate.strftime("%Y%m%d")
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'sss',
            'standard_name': 'sea_surface_salinity',
            'long_name': 'sea_surface_salinity',
            'level': "SURFACE",
            'units': "psu",

            'grid': {
                'type': "LatLon",
                'name': "RTOFS Grid",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs

if file_flag == 'obs':
    met_data = obs2[:,:]
    #trim the lat/lon grids so they match the data fields
    lat_met = obs2.lat
    lon_met = obs2.lon
    v_str = vDate.strftime("%Y%m%d")
```

(continues on next page)

(continued from previous page)

```
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'sss',
            'standard_name': 'analyzed sea surface salinity',
            'long_name': 'sea_surface_salinity',
            'level': "SURFACE",
            'units': "psu",

            'grid': {
                'type': "LatLon",
                'name': "Lat Lon",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs

if file_flag == 'climo':
    met_data = climo2[:,:]
    #modify the lat and lon grids since they need to match the data dimensions, and code␣
↪cuts the last row/column of data
    lat_met = climo2.lat
    lon_met = climo2.lon
    v_str = vDate.strftime("%Y%m%d")
    v_str = v_str + '_000000'
    lat_ll = float(lat_met.min())
    lon_ll = float(lon_met.min())
    n_lat = lat_met.shape[0]
```

(continues on next page)

(continued from previous page)

```
    n_lon = lon_met.shape[0]
    delta_lat = (float(lat_met.max()) - float(lat_met.min()))/float(n_lat)
    delta_lon = (float(lon_met.max()) - float(lon_met.min()))/float(n_lon)
    print(f"variables:"
            f"lat_ll: {lat_ll} lon_ll: {lon_ll} n_lat: {n_lat} n_lon: {n_lon} delta_lat:
↪{delta_lat} delta_lon: {delta_lon}")
    met_data.attrs = {
            'valid': v_str,
            'init': v_str,
            'lead': "00",
            'accum': "00",
            'name': 'sea_water_salinity',
            'standard_name': 'sea_water_salinity',
            'long_name': 'sea_water_salinity',
            'level': "SURFACE",
            'units': "psu",

            'grid': {
                'type': "LatLon",
                'name': "crs Grid",
                'lat_ll': lat_ll,
                'lon_ll': lon_ll,
                'delta_lat': delta_lat,
                'delta_lon': delta_lon,
                'Nlat': n_lat,
                'Nlon': n_lon,
                }
            }
    attrs = met_data.attrs
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstRTOFS_obsSMAP_climWOA_sss.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstRTOFS_obsSMAP_climWOA_sss.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/marine_and_
↪cryosphere/GridStat_fcstRTOFS_obsSMAP_climWOA_sss.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for thisIce use case will be found in 20210503 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_SSS_000000L_20210502_000000V.stat
- grid_stat_SSS_000000L_20210502_000000V_cnt.txt
- grid_stat_SSS_000000L_20210502_000000V_pairs.nc

Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase
- ClimatologyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/marine_and_cryosphere-GridStat_fcstRTOFS_obsSMAP_climWOA_sss.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.7 Medium Range

Lower resolution model configuration (>4km) usually producing forecasts out to 7-14 days (also referred to as global models)

7.2.16.7.1 Multi_Tool: Feature Relative by Lead (with lead groupings)

model_applicaitons/medium_range/ TCStat_SeriesAnalysis_fcstGFS _obsGFS_FeatureRelative _Series-ByLead.conf

Scientific Objective

By maintaining focus of each evaluation time (or evaluation time series, in this case) on a user-defined area around a cyclone, the model statistical errors associated with cyclonic physical features (moisture flux, stability, strength of upper-level PV anomaly and jet, etc.) can be related directly to the model forecasts and provide improvement guidance by accurately depicting interactions with significant weather features around and within the cyclone. This is in contrast to the traditional method of regional averaging cyclone observations in a fixed grid, which "smooths out" system features and limits the meaningful metrics that can be gathered. Specifically, this use case creates bins of forecast lead times as specified by the given ranges which provides additional insight directly into forecast lead time accuracy.

Datasets

Relevant information about the datasets that would be beneficial include:

- TC-Pairs/TC-Stat Forecast dataset: ADeck modified-ATCF tropical cyclone data
- Series-Analysis Forecast dataset: GFS
- TC-Pairs/TC-Stat Observation dataset: BDeck modified-ATCF tropical cyclone data
- Series-Analysis Observation dataset: GFS Analysis

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* netCDF4
```

METplus Components

This use case first runs TCPairs and ExtractTiles wrappers to generate matched tropical cyclone data and regrid them into appropriately-sized tiles along a storm track. The MET tc-stat tool is used to filter the track data, and the MET regrid-dataplane tool is used to regrid the data (GRIB1 or GRIB2 into netCDF). Next, a series analysis by lead time is performed on the results and plots (.ps and .png) are generated for all variable-level-stat combinations from the specified variables, levels, and requested statistics. The final results are aggregated into forecast hour groupings as specified by the start, end and increment in the METplus configuration file, as well as labels to identify each forecast hour grouping.

METplus Workflow

The following tools are used for each run time:

TCPairs > RegridDataPlane, TCStat > SeriesAnalysis

This example loops by forecast/lead time (with begin, end, and increment as specified in the METplus TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.conf file). The following will be run based on the availability of data corresponding to the initialization time (in this example, we only have 20141214 as our initialization time) and the requested forecast leads, resulting in the run times below.

Run times:

Init: 20141214_0Z
Forecast lead: 6, 12, 18, 24, 30, 36, 42

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesB

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/medium_range/TCStat_
```

(continues on next page)

(continued from previous page)

```
↪SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCPairs, TCStat, ExtractTiles, TCStat(for_series_analysis), SeriesAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20141214
INIT_END = 20141214
INIT_INCREMENT = 21600 ;; set to every 6 hours=21600 seconds

LEAD_SEQ_1 = begin_end_incr(0,18,6)
LEAD_SEQ_1_LABEL = Day1

LEAD_SEQ_2 = begin_end_incr(24,42,6)
LEAD_SEQ_2_LABEL = Day2


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/track_data
```

(continues on next page)

(continued from previous page)

```
TC_PAIRS_ADECK_TEMPLATE = {date?fmt=%Y%m}/a{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}

TC_PAIRS_BDECK_INPUT_DIR = {TC_PAIRS_ADECK_INPUT_DIR}
TC_PAIRS_BDECK_TEMPLATE = {date?fmt=%Y%m}/b{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}

TC_PAIRS_REFORMAT_DIR = {OUTPUT_BASE}/track_data_atcf
TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs
TC_PAIRS_OUTPUT_TEMPLATE = {date?fmt=%Y%m}/{basin?fmt=%s}q{date?fmt=%Y%m%d%H}.gfso.{cyclone?
↪fmt=%s}

TC_PAIRS_SKIP_IF_REFORMAT_EXISTS = yes
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = yes


TC_STAT_LOOKIN_DIR = {TC_PAIRS_OUTPUT_DIR}
TC_STAT_OUTPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
TC_STAT_DUMP_ROW_TEMPLATE = filter_{init?fmt=%Y%m%d_%H}.tcst


EXTRACT_TILES_TC_STAT_INPUT_DIR = {TC_STAT_OUTPUT_DIR}
EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}

EXTRACT_TILES_GRID_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/reduced_model_
↪data

FCST_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/reduced_model_
↪data
FCST_EXTRACT_TILES_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}
↪00_{lead?fmt=%3H}.grb2

OBS_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/reduced_model_data
OBS_EXTRACT_TILES_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}
↪00_000.grb2

EXTRACT_TILES_OUTPUT_DIR = {OUTPUT_BASE}/extract_tiles
FCST_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/{storm_id}/FCST_TILE_F{lead?fmt=
↪%3H}_gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%3H}.nc
OBS_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/{storm_id}/OBS_TILE_F{lead?fmt=%3H}_
↪gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%3H}.nc

EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS = yes
```

(continues on next page)

(continued from previous page)

```
FCST_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = {FCST_EXTRACT_TILES_OUTPUT_TEMPLATE}

OBS_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = {OBS_EXTRACT_TILES_OUTPUT_TEMPLATE}

SERIES_ANALYSIS_TC_STAT_INPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}
SERIES_ANALYSIS_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/series_analysis_lead
SERIES_ANALYSIS_OUTPUT_TEMPLATE = {label}/series_F{fcst_beg}_to_F{fcst_end}_{fcst_name}_
↪{fcst_level}.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = GFSO

BOTH_VAR1_NAME = TMP
BOTH_VAR1_LEVELS = Z2


###
# TCPairs Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcpairs
###

TC_PAIRS_SKIP_LEAD_SEQ = True

TC_PAIRS_INIT_INCLUDE =
TC_PAIRS_INIT_EXCLUDE =

TC_PAIRS_VALID_BEG =
TC_PAIRS_VALID_END =

TC_PAIRS_STORM_ID =
TC_PAIRS_BASIN =
TC_PAIRS_CYCLONE =
TC_PAIRS_STORM_NAME =

TC_PAIRS_DLAND_FILE = {MET_INSTALL_DIR}/share/met/tc_data/dland_global_tenth_degree.nc
```

(continues on next page)

(continued from previous page)

```
TC_PAIRS_REFORMAT_DECK = yes
TC_PAIRS_REFORMAT_TYPE = SBU

TC_PAIRS_MISSING_VAL_TO_REPLACE = -99
TC_PAIRS_MISSING_VAL = -9999


###
# TCStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcstat
###

TC_STAT_JOB_ARGS = -job filter -basin ML -dump_row {TC_STAT_OUTPUT_DIR}/{TC_STAT_DUMP_ROW_
↪TEMPLATE}

TC_STAT_MATCH_POINTS = true

TC_STAT_AMODEL =
TC_STAT_BMODEL =
TC_STAT_DESC =
TC_STAT_STORM_ID =
TC_STAT_BASIN =
TC_STAT_CYCLONE =
TC_STAT_STORM_NAME =

TC_STAT_INIT_BEG =
TC_STAT_INIT_END =
TC_STAT_INIT_INCLUDE = {init?fmt=%Y%m%d_%H}
TC_STAT_INIT_EXCLUDE =
TC_STAT_INIT_HOUR =

TC_STAT_VALID_BEG =
TC_STAT_VALID_END =
TC_STAT_VALID_INCLUDE =
TC_STAT_VALID_EXCLUDE =
TC_STAT_VALID_HOUR =
TC_STAT_LEAD_REQ =
TC_STAT_INIT_MASK =
TC_STAT_VALID_MASK =

TC_STAT_VALID_HOUR =
TC_STAT_LEAD =

TC_STAT_TRACK_WATCH_WARN =
```

(continues on next page)

(continued from previous page)

```
TC_STAT_COLUMN_THRESH_NAME =
TC_STAT_COLUMN_THRESH_VAL =

TC_STAT_COLUMN_STR_NAME =
TC_STAT_COLUMN_STR_VAL =

TC_STAT_INIT_THRESH_NAME =
TC_STAT_INIT_THRESH_VAL =

TC_STAT_INIT_STR_NAME =
TC_STAT_INIT_STR_VAL =

TC_STAT_WATER_ONLY =

TC_STAT_LANDFALL =

TC_STAT_LANDFALL_BEG =
TC_STAT_LANDFALL_END =


###
# ExtractTiles Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#extracttiles
###

EXTRACT_TILES_NLAT = 60
EXTRACT_TILES_NLON = 60

EXTRACT_TILES_DLAT = 0.5
EXTRACT_TILES_DLON = 0.5

EXTRACT_TILES_LON_ADJ = 15
EXTRACT_TILES_LAT_ADJ = 15


###
# TCStat (for SeriesAnalysis) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcstat
###

# Settings specific to the TCStat(for_series_analysis) process that was set
# in the PROCESS_LIST. Any TC_STAT_* variable not set in this section will use
# the value set outside of this section
[for_series_analysis]
```

(continues on next page)

(continued from previous page)

```
TC_STAT_JOB_ARGS = -job filter -init_beg {INIT_BEG} -init_end {INIT_END} -dump_row {TC_STAT_
↪OUTPUT_DIR}/{TC_STAT_DUMP_ROW_TEMPLATE}

TC_STAT_OUTPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}
TC_STAT_LOOKIN_DIR = {EXTRACT_TILES_OUTPUT_DIR}


###
# SeriesAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#seriesanalysis
###

[config]

SERIES_ANALYSIS_BACKGROUND_MAP = no

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False

SERIES_ANALYSIS_REGRID_TO_GRID = FCST
SERIES_ANALYSIS_REGRID_METHOD = FORCE

SERIES_ANALYSIS_STAT_LIST = TOTAL, FBAR, OBAR, ME

SERIES_ANALYSIS_BLOCK_SIZE = 4000

SERIES_ANALYSIS_IS_PAIRED = True

SERIES_ANALYSIS_GENERATE_PLOTS = yes

SERIES_ANALYSIS_GENERATE_ANIMATIONS = yes

PLOT_DATA_PLANE_TITLE = {MODEL} series_F{fcst_beg}_to_F{fcst_end} Forecasts{nseries}, {stat}␣
↪for {fcst_name} {fcst_level}
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

TCPairsConfig_wrapped

Note: See the *TCPairs MET Configuration* (page 282) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

//
// Models
//
${METPLUS_MODEL}

//
// Description
//
${METPLUS_DESC}

//
// Storm identifiers
//
${METPLUS_STORM_ID}

//
// Basins
//
${METPLUS_BASIN}

//
// Cyclone numbers
//
${METPLUS_CYCLONE}

//
// Storm names
//
${METPLUS_STORM_NAME}
```

(continues on next page)

(continued from previous page)

```
//
// Model initialization time windows to include or exclude
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
${METPLUS_VALID_INC}
// valid_exc =
${METPLUS_VALID_EXC}

// write_valid =
${METPLUS_WRITE_VALID}

//
// Valid model time window
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}

//
// Model initialization hours
//
init_hour = [];

//
// Required lead time in hours
//
lead_req = [];

//
// lat/lon polylines defining masking regions
//
init_mask  = "";
valid_mask = "";

//
// Specify if the code should check for duplicate ATCF lines
//
//check_dup =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CHECK_DUP}


//
// Specify special processing to be performed for interpolated models.
// Set to NONE, FILL, or REPLACE.
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
//
//consensus =
${METPLUS_CONSENSUS_LIST}


//
// Forecast lag times
//
lag_time = [];

//
// CLIPER/SHIFOR baseline forecasts to be derived from the BEST
// and operational (CARQ) tracks.
//
best_technique = [ "BEST" ];
best_baseline  = [];
oper_technique = [ "CARQ" ];
oper_baseline  = [];

//
// Specify the datasets to be searched for analysis tracks (NONE, ADECK, BDECK,
// or BOTH).
//
anly_track = BDECK;

//
// Specify if only those track points common to both the ADECK and BDECK
// tracks be written out.
//
//match_points =
${METPLUS_MATCH_POINTS}

//
```

(continues on next page)

(continued from previous page)

```
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
${METPLUS_DLAND_FILE}

//
// Specify watch/warning information:
//   - Input watch/warning filename
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}


//diag_info_map = {
${METPLUS_DIAG_INFO_MAP_LIST}

//diag_convert_map = {
${METPLUS_DIAG_CONVERT_MAP_LIST}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

TCStatConfig_wrapped

Note: See the *TCStat MET Configuration* (page 297) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCStat configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// The parameters listed below are used to filter the TC-STAT data down to the
// desired subset of lines over which statistics are to be computed.  Only
// those lines which meet ALL of the criteria specified will be retained.
//
// The settings that are common to all jobs may be specified once at the top
// level.  If no selection is listed for a parameter, that parameter will not
// be used for filtering.  If multiple selections are listed for a parameter,
// the analyses will be performed on their union.
//

//
// Stratify by the AMODEL or BMODEL columns.
//
${METPLUS_AMODEL}
${METPLUS_BMODEL}

//
// Stratify by the DESC column.
//
${METPLUS_DESC}

//
// Stratify by the STORM_ID column.
//
${METPLUS_STORM_ID}

//
// Stratify by the BASIN column.
// May add using the "-basin" job command option.
//
${METPLUS_BASIN}

//
// Stratify by the CYCLONE column.
// May add using the "-cyclone" job command option.
//
${METPLUS_CYCLONE}

//
// Stratify by the STORM_NAME column.
// May add using the "-storm_name" job command option.
//
${METPLUS_STORM_NAME}

//
```

(continues on next page)

(continued from previous page)

```
// Stratify by the INIT times.
// Model initialization time windows to include or exclude
// May modify using the "-init_beg", "-init_end", "-init_inc",
// and "-init_exc" job command options.
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
${METPLUS_INIT_INC}
${METPLUS_INIT_EXC}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INC}
${METPLUS_VALID_EXC}

//
// Stratify by the initialization and valid hours and lead time.
//
${METPLUS_INIT_HOUR}
${METPLUS_VALID_HOUR}
${METPLUS_LEAD}

//
// Select tracks which contain all required lead times.
//
${METPLUS_LEAD_REQ}

//
// Stratify by the INIT_MASK and VALID_MASK columns.
//
${METPLUS_INIT_MASK}
${METPLUS_VALID_MASK}


//
// Stratify by the LINE_TYPE column.
//
//line_type =
${METPLUS_LINE_TYPE}


//
```

(continues on next page)

(continued from previous page)

```
// Stratify by checking the watch/warning status for each track point
// common to both the ADECK and BDECK tracks.  If the watch/warning status
// of any of the track points appears in the list, retain the entire track.
//
${METPLUS_TRACK_WATCH_WARN}

//
// Stratify by applying thresholds to numeric data columns.
//
${METPLUS_COLUMN_THRESH_NAME}
${METPLUS_COLUMN_THRESH_VAL}

//
// Stratify by performing string matching on non-numeric data columns.
//
${METPLUS_COLUMN_STR_NAME}
${METPLUS_COLUMN_STR_VAL}

//
// Stratify by excluding strings in non-numeric data columns.
//
//column_str_exc_name =
${METPLUS_COLUMN_STR_EXC_NAME}

//column_str_exc_val =
${METPLUS_COLUMN_STR_EXC_VAL}

//
// Similar to the column_thresh options above
//
${METPLUS_INIT_THRESH_NAME}
${METPLUS_INIT_THRESH_VAL}

//
// Similar to the column_str options above
//
${METPLUS_INIT_STR_NAME}
${METPLUS_INIT_STR_VAL}

//
// Similar to the column_str_exc options above
//
//init_str_exc_name =
${METPLUS_INIT_STR_EXC_NAME}
```

(continues on next page)

(continued from previous page)

```
//init_str_exc_val =
${METPLUS_INIT_STR_EXC_VAL}

//diag_thresh_name =
${METPLUS_DIAG_THRESH_NAME}

//diag_thresh_val =
${METPLUS_DIAG_THRESH_VAL}

//init_diag_thresh_name =
${METPLUS_INIT_DIAG_THRESH_NAME}

//init_diag_thresh_val =
${METPLUS_INIT_DIAG_THRESH_VAL}


//
// Stratify by the ADECK and BDECK distances to land.
//
${METPLUS_WATER_ONLY}

//
// Specify whether only those track points occurring near landfall should be
// retained, and define the landfall retention window in HH[MMSS] format
// around the landfall time.
//
${METPLUS_LANDFALL}
${METPLUS_LANDFALL_BEG}
${METPLUS_LANDFALL_END}

//
// Specify whether only those track points common to both the ADECK and BDECK
// tracks should be retained.  May modify using the "-match_points" job command
// option.
//
${METPLUS_MATCH_POINTS}

//event_equal =
${METPLUS_EVENT_EQUAL}

//event_equal_lead =
${METPLUS_EVENT_EQUAL_LEAD}

//out_init_mask =
${METPLUS_OUT_INIT_MASK}
```

(continues on next page)

(continued from previous page)

```
//out_valid_mask =
${METPLUS_OUT_VALID_MASK}


//
// Array of TCStat analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

SeriesAnalysisConfig_wrapped

Note: See the *SeriesAnalysis MET Configuration* (page 232) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Series-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
//
//desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
//obtype =
${METPLUS_OBTYPE}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

censor_thresh = [];
censor_val    = [];
//cat_thresh =
${METPLUS_CAT_THRESH}
cnt_thresh    = [ NA ];
cnt_logic     = UNION;

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_CAT_THRESH}
   ${METPLUS_FCST_FIELD}
}
obs = {
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}
```

(continues on next page)

(continued from previous page)

```
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

//
// Number of grid points to be processed concurrently.  Set smaller to use
// less memory but increase the number of passes through the data.
//
//block_size =
${METPLUS_BLOCK_SIZE}

//
// Ratio of valid matched pairs to compute statistics for a grid point
//
//vld_thresh =
${METPLUS_VLD_THRESH}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_stats = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_STATS_DICT}

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.conf, then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.conf
-c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

  and for the [exe] section, you will need to define the location of NON-MET executables. If the executable is in the user's path, METplus will find it from the name. If the executable is not in

the path, specify the full path to the executable here (i.e. CONVERT = /usr/bin/convert) The following executables are required for performing series analysis use cases:

If the executables are in the path:

- **CONVERT = convert**

NOTE: All of these executable items must be located under the [exe] section.

If the executables are not in the path, they need to be defined:

- **CONVERT = /path/to/convert**

NOTE: All of these executable items must be located under the [exe] section. Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y

[exe]
CONVERT = /path/to/convert
```

NOTE: The INPUT_BASE, OUTPUT_BASE, and MET_INSTALL_DIR must be located under the [dir] section, while the RM, CUT, TR, NCAP2, CONVERT, and NCDUMP must be located under the [exe] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in series_analysis_lead, relative to the **OUTPUT_BASE**, and in the following directories (relative to **OUTPUT_BASE**):

- Day1
- Day2
- series_animate

The *Day1* subdirectory will contain files that have the following format:

ANLY_FILES_Fhhh_to_FHHH

FCST_ASCII_FILES_Fhhh_to_FHHH

series_<varname>_<level>_<stat>.png

series_<varname>_<level>_<stat>.ps

series_<varname>_<level>_<stat>.nc

Where:

hhh is the starting forecast hour/lead time in hours

HHH is the ending forecast hour/lead time in hours

varname is the variable of interest, as specified in the METplus series_by_lead_all_fhrs config file

level is the level of interest, as specified in the METplus series_by_lead_all_fhrs config file

stat is the statistic of interest, as specified in the METplus series_by_lead_all_fhrs config file.

The *Day2* subdirectory will contain files that have the same formatting as *Day1*, but for those forecast hours within 24 to 42 hours.

The series_animate directory contains the animations of the series analysis in .gif format for all variable, level, and statistics combinations:

series_animate_<varname>_<level>_<stat>.gif

Keywords

Note:

- MediumRangeAppUseCase
- TCPairsToolUseCase
- SeriesByLeadUseCase
- TCStatToolUseCase
- RegridDataPlaneToolUseCase
- MediumRangeAppUseCase
- SeriesAnalysisUseCase
- GRIB2FileUseCase
- FeatureRelativeUseCase
- SBUOrgUseCase
- DiagnosticsUseCase
- RuntimeFreqUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/medium_range-TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_Se

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.7.2 Grid-Stat: Standard Verification of Surface Fields

model_applications/medium_range/GridStat_fcstGFS_obsGFS _Sfc_MultiField.conf

Scientific Objective

To provide useful statistical information on the relationship between observation data in gridded format to a gridded forecast. These values can be used to assess the skill of the prediction. Statistics stored only as partial sums to save space. Stat-Analysis must be used to compute Continuous Statistics.

Datasets

Forecast: GFS
Observation: GFS
Location: Click here for the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1244) section for more information.
Data Source: GFS

METplus Components

This use case utilizes the METplus GridStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool grid_stat if all required files are found.

METplus Workflow

GridStat is the only tool called in this example. It processes the following run times:

Valid: 2017-06-13 0Z
Forecast lead: 24 hour

Valid: 2017-06-13 6Z

Forecast lead: 24 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/medium_range/GridStat_fcstGFS_obsGFS_Sfc_MultiField.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/medium_range/
↪GridStat_fcstGFS_obsGFS_Sfc_MultiField.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2017061300
VALID_END = 2017061306
VALID_INCREMENT = 21600

LEAD_SEQ = 24
```

(continues on next page)

(continued from previous page)

```
###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_grid/gfs/fcst
FCST_GRID_STAT_INPUT_TEMPLATE = pgbf{lead?fmt=%HHH}.gfs.{init?fmt=%Y%m%d%H}

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_grid/gfs/fcst
OBS_GRID_STAT_INPUT_TEMPLATE = pgbf000.gfs.{valid?fmt=%Y%m%d%H}

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_out/{MODEL}/sfc
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H%M}/grid_stat


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

BOTH_VAR1_NAME = TMP
FCST_VAR1_OPTIONS = GRIB_lvl_typ = 105;
BOTH_VAR1_LEVELS = Z2

BOTH_VAR2_NAME = RH
FCST_VAR2_OPTIONS = GRIB_lvl_typ = 105;
BOTH_VAR2_LEVELS = Z2

BOTH_VAR3_NAME = SPFH
FCST_VAR3_OPTIONS = GRIB_lvl_typ = 105;
BOTH_VAR3_LEVELS = Z2

BOTH_VAR4_NAME = HPBL
FCST_VAR4_OPTIONS = GRIB_lvl_typ = 01;
BOTH_VAR4_LEVELS = L0

BOTH_VAR5_NAME = PRES
FCST_VAR5_OPTIONS = GRIB_lvl_typ = 01;
BOTH_VAR5_LEVELS = Z0

BOTH_VAR6_NAME = PRMSL
FCST_VAR6_OPTIONS = GRIB_lvl_typ = 102;
BOTH_VAR6_LEVELS = L0
```

(continues on next page)

(continued from previous page)

```
BOTH_VAR7_NAME = TMP
FCST_VAR7_OPTIONS = GRIB_lvl_typ = 01;
BOTH_VAR7_LEVELS = Z0

BOTH_VAR8_NAME = UGRD
FCST_VAR8_OPTIONS = GRIB_lvl_typ = 105;
BOTH_VAR8_LEVELS = Z10

BOTH_VAR9_NAME = VGRD
FCST_VAR9_OPTIONS = GRIB_lvl_typ = 105;
BOTH_VAR9_LEVELS = Z10

BOTH_VAR10_NAME = TSOIL
FCST_VAR10_OPTIONS = GRIB_lvl_typ = 112;
BOTH_VAR10_LEVELS = Z0-10

BOTH_VAR11_NAME = SOILW
FCST_VAR11_OPTIONS = GRIB_lvl_typ = 112;
BOTH_VAR11_LEVELS = Z0-10

BOTH_VAR12_NAME = WEASD
FCST_VAR12_OPTIONS = GRIB_lvl_typ = 01;
BOTH_VAR12_LEVELS = Z0

BOTH_VAR13_NAME = CAPE
FCST_VAR13_OPTIONS = GRIB_lvl_typ = 01;
BOTH_VAR13_LEVELS = Z0

BOTH_VAR14_NAME = CWAT
FCST_VAR14_OPTIONS = GRIB_lvl_typ = 200;
BOTH_VAR14_LEVELS = L0

BOTH_VAR15_NAME = PWAT
FCST_VAR15_OPTIONS = GRIB_lvl_typ = 200;
BOTH_VAR15_LEVELS = L0

BOTH_VAR16_NAME = TMP
FCST_VAR16_OPTIONS = GRIB_lvl_typ = 07;
BOTH_VAR16_LEVELS = L0

BOTH_VAR17_NAME = HGT
FCST_VAR17_OPTIONS = GRIB_lvl_typ = 07;
BOTH_VAR17_LEVELS = L0
```

(continues on next page)

(continued from previous page)

```
BOTH_VAR18_NAME = TOZNE
FCST_VAR18_OPTIONS = GRIB_lvl_typ = 200;
BOTH_VAR18_LEVELS = L0


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

MODEL = GFS
OBTYPE = ANLYS

GRID_STAT_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}_
↪{CURRENT_FCST_LEVEL}

GRID_STAT_REGRID_TO_GRID = G002
GRID_STAT_REGRID_METHOD = BILIN
GRID_STAT_REGRID_WIDTH = 2

GRID_STAT_MASK_POLY = {INPUT_BASE}/model_applications/medium_range/poly/NHX.nc,
 {INPUT_BASE}/model_applications/medium_range/poly/SHX.nc,
 {INPUT_BASE}/model_applications/medium_range/poly/N60.nc,
 {INPUT_BASE}/model_applications/medium_range/poly/S60.nc,
 {INPUT_BASE}/model_applications/medium_range/poly/TRO.nc,
 {INPUT_BASE}/model_applications/medium_range/poly/NPO.nc,
 {INPUT_BASE}/model_applications/medium_range/poly/SPO.nc,
 {INPUT_BASE}/model_applications/medium_range/poly/NAO.nc,
 {INPUT_BASE}/model_applications/medium_range/poly/SAO.nc,
 {INPUT_BASE}/model_applications/medium_range/poly/CONUS.nc,
 {INPUT_BASE}/model_applications/medium_range/poly/CAM.nc,
 {INPUT_BASE}/model_applications/medium_range/poly/NSA.nc

GRID_STAT_CLIMO_CDF_WRITE_BINS = False

GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE

GRID_STAT_GRID_WEIGHT_FLAG = COS_LAT
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST

GRID_STAT_CLIMO_MEAN_REGRID_METHOD = BILIN
GRID_STAT_CLIMO_MEAN_REGRID_WIDTH = 2
GRID_STAT_CLIMO_MEAN_DAY_INTERVAL = 1
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
```

(continues on next page)

(continued from previous page)

```
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstGFS_obsGFS_Sfc_MultiField.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪GridStat_fcstGFS_obsGFS_Sfc_MultiField.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstGFS_obsGFS_Sfc_MultiField.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪GridStat_fcstGFS_obsGFS_Sfc_MultiField.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in met_out/{MODEL}/sfc (relative to **OUTPUT_BASE**) and will contain the following files:

- 00Z/GFS/GFS_20170613.stat
- 06Z/GFS/GFS_20170613.stat

Keywords

Note:

- GridStatToolUseCase
- MediumRangeAppUseCase
- GRIBFileUseCase
- NOAAEMCOrgUseCase
- ClimatologyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/medium_range-GridStat_fcstGFS_obsGFS_Sfc_MultiField.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.7.3 GridStat: Use binary observation field to verify percentile forecast

model_applications/medium_range/GridStat_fcstGEFS_obsCADB_BinaryObsPOE.conf

Scientific Objective

Evaluation of a Probability of Exceedence (POE) field presents several difficulties. Some of these include a fitting verification statistic to report on, choosing a meaningful percentile field, and more. This use case was the culmination of attempting to verify a POE field for extreme temperature (defined as the 85th percentile) in METplus. In order to provide a streamlined process that didn't require vast reworkings of the MET tools, the observation field was converted to binary: 0s indicating a non-85th percentile temperature was observed, and a 1 indicating the opposite. Those observations are compared to the chosen forecast percentile and the HSS_EC becomes the main statistical focus, as the new hss_ec_value feature allowed the use case to more closely replicate in-house verificaiton that already existed. A final note that because the POE forecast file is a non-standard netCDF, Python Embedding was used to extract the desired field

Datasets

Forecast: 85th percentile of Temperature maximum, from GEFS

Observations: Climate Assessment Data Base (CADB), converted into a binary field relative to the 85th percentile

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1256) section for more information.

Data Source: CPC

METplus Components

This use case calls a Python script to extract the user-defined percentile forecast. METplus then verifies it against a binary observation field in GridStat and returns the requested line type outputs.

METplus Workflow

The following boundary time is used for the entire script:

Init Beg: 2022-05-22
Init End: 2022-05-22

There is only one time processed for the use case.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. -c parm/use_cases/model_applications/medium_range/GridStat_fcstGEFS_obsCADB_BinaryObsPOE.conf

```
[config]


PROCESS_LIST = GridStat


LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG=20220522
INIT_END=20220522
INIT_INCREMENT = 12H

LEAD_SEQ = 8d
```

(continues on next page)

(continued from previous page)

```
###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/GridStat_fcstGEFS_
↪obsCADB_BinaryObsPOE
OBS_GRID_STAT_INPUT_TEMPLATE = tmax_cats_{valid?fmt=%Y%m%d}.nc

GRID_STAT_CLIMO_MEAN_INPUT_DIR =
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/POE_tmax
GRID_STAT_OUTPUT_TEMPLATE =


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = GEFS
OBTYPE = Obs

GRID_STAT_ONCE_PER_FIELD = False

FCST_IS_PROB = false
#FCST_GRID_STAT_PROB_THRESH = ==0.1

FCST_VAR1_NAME = {PARM_BASE}/use_cases/model_applications/medium_range/GridStat_fcstGEFS_
↪obsCADB_BinaryObsPOE/Tmax_fcst_embedded.py {INPUT_BASE}/model_applications/medium_range/
↪GridStat_fcstGEFS_obsCADB_BinaryObsPOE/gefs-00z_rfcst-cal_tmax_{init?fmt=%Y%m%d}_day8.nc:85
FCST_VAR1_THRESH = >=0.2

OBS_VAR1_NAME = tmax
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_THRESH = gt0.5
```

(continues on next page)

(continued from previous page)

```
###
# GridStat Settings (optional)
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

#LOG_GRID_STAT_VERBOSITY = 2

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

#FCST_GRID_STAT_FILE_TYPE =
OBS_GRID_STAT_FILE_TYPE = NETCDF_NCCF

GRID_STAT_REGRID_TO_GRID = FCST


GRID_STAT_DESC = NA

FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0
OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_OUTPUT_PREFIX =

#GRID_STAT_OUTPUT_FLAG_FHO = NONE
GRID_STAT_OUTPUT_FLAG_CTC = BOTH
GRID_STAT_OUTPUT_FLAG_CTS = BOTH
#GRID_STAT_OUTPUT_FLAG_MCTC = NONE
#GRID_STAT_OUTPUT_FLAG_MCTS = NONE
#GRID_STAT_OUTPUT_FLAG_CNT = NONE
#GRID_STAT_OUTPUT_FLAG_SL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_SAL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_VL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_VAL1L2 = NONE
#GRID_STAT_OUTPUT_FLAG_VCNT = NONE
#GRID_STAT_OUTPUT_FLAG_PCT = NONE
#GRID_STAT_OUTPUT_FLAG_PSTD = NONE
#GRID_STAT_OUTPUT_FLAG_PJC = NONE
```

(continues on next page)

(continued from previous page)

```
#GRID_STAT_OUTPUT_FLAG_PRC = NONE
#GRID_STAT_OUTPUT_FLAG_ECLV = BOTH
#GRID_STAT_OUTPUT_FLAG_NBRCTC = NONE
#GRID_STAT_OUTPUT_FLAG_NBRCTS = NONE
#GRID_STAT_OUTPUT_FLAG_NBRCNT = NONE
#GRID_STAT_OUTPUT_FLAG_GRAD = BOTH
#GRID_STAT_OUTPUT_FLAG_DMAP = NONE

GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
#GRID_STAT_NC_PAIRS_FLAG_CLIMO_CDP = FALSE
#GRID_STAT_NC_PAIRS_FLAG_WEIGHT = FALSE
#GRID_STAT_NC_PAIRS_FLAG_NBRHD = FALSE
#GRID_STAT_NC_PAIRS_FLAG_FOURIER = FALSE
#GRID_STAT_NC_PAIRS_FLAG_GRADIENT = FALSE
#GRID_STAT_NC_PAIRS_FLAG_DISTANCE_MAP = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE

GRID_STAT_HSS_EC_VALUE = 0.15

#GRID_STAT_MASK_GRID =
GRID_STAT_MASK_POLY = {MET_INSTALL_DIR}/share/met/poly/CONUS.poly, {MET_INSTALL_DIR}/share/
↪met/poly/EAST.poly, {MET_INSTALL_DIR}/share/met/poly/WEST.poly, {MET_INSTALL_DIR}/share/
↪met/poly/NPL.poly, {MET_INSTALL_DIR}/share/met/poly/NEC.poly
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
```

(continues on next page)

(continued from previous page)

```
rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
```

(continues on next page)

(continued from previous page)

```
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}
```

(continues on next page)

(continued from previous page)

```
tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case calls a Python script to parse the user-requested percentile from the forecast dataset. This is controlled in the forecast VAR1 variable setting and is provided in parm/use_cases/model_applications/medium_range/GridStat_fcstGEFS_obsCADB_BinaryObsPOE/Tmax_fcst_embedd

```
import sys
import datetime as dt
import numpy as np
from netCDF4 import Dataset


try:
    #user will input name of the file, as well as a percentile they're interested in
    #in future iteration, this may need to change to multiple percentiles (a la list style)
    print("1")
    input_file,ptile = sys.argv[1].split(':')
    f = Dataset(input_file, 'r')
    print("2")
    v = f['poe']
    val_time = f.valid_date_range[1]
    val_time = dt.datetime.strptime(val_time,"%Y%m%d")
    ini_time = str(input_file.split('_')[-2])
    ini_time = dt.datetime.strptime(ini_time,"%Y%m%d")
    print("3")
    lead, rem = divmod((val_time - ini_time).total_seconds(), 3600)
    ptile_ind = np.where(f['ptile'][:] == int(ptile))[0][0]
    print("4")
    lat = np.float64(f.variables['latitude'][:])
    lon = np.float64(f.variables['longitude'][:])
    #var = np.float64(v[0,ptile_ind,:,:],fill_value=-9999.)
    var = np.float64(v[0,ptile_ind,:,:])
    print(np.amax(var),np.amin(var))
    met_data = var.copy()
except NameError:
```

(continues on next page)

(continued from previous page)

```
    print("Can't find input file")
    sys.exit(1)

#ADDED
#for i in range(len(met_data)):
#    for j in range(len(met_data[i])):
#        if j <=2 or j >=358:
#            print("edge of ", met_data[i,j],"at", lat[i],lon[j])
#        if lat[i] >=42.0 and lat[i] <= 46.0:
#            if lon[j] >= 235.0 and lon[j] <= 239.0:
#                print("found",met_data[i,j]," at ",lat[i],lon[j],i,j)

attrs = {

        'valid': str(val_time.strftime("%Y%m%d"))+'_000000',
        'init': str(ini_time.strftime("%Y%m%d"))+'_000000',
        'name': 'poe_P'+str(ptile),
        'long_name': v.long_name,
        'lead': str(int(lead)),
        'accum': '00',
        'level': 'SURFACE',
        'units': 'PERCENTILES',

        'grid': {
            'name': 'Global 1 degree',
            'type': 'LatLon',
            'lat_ll': -90.0,
            'lon_ll': 0.0,
            'delta_lat': 1.0,
            'delta_lon': 1.0,

            'Nlon': f.dimensions['longitude'].size,
            'Nlat': f.dimensions['latitude'].size,
            }
        }

#print output for user to show successful run
print("Input file: " + repr(input_file.split('/')[-1]))
print("Attributes:\t"+ repr(attrs))
f.close()
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstGEFS_obsCADB_BinaryObsPOE.conf then a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/medium_range/GridStat_
↪fcstGEFS_obsCADB_BinaryObsPOE.conf /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstGEFS_obsCADB_BinaryObsPOE:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/marine_and_cryosphere/
↪GridStat_fcstGEFS_obsCADB_BinaryObsPOE.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[config]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for the use case will be found in model_applications/POE_tmax (relative to **OUTPUT_BASE**). The following files should exist:

- grid_stat_1920000L_20220530_000000V_ctc.txt
- grid_stat_1920000L_20220530_000000V_cts.txt
- grid_stat_1920000L_20220530_000000V.stat

Keywords

Note:

- GridStatUseCase
- PythonEmbeddingFileUseCase
- MediumRangeAppUseCase
- NETCDFFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/medium_range-GridStat_fcstGEFS_obsCADB_BinaryObsPOE.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.7.4 Multi_Tool: Feature Relative by Init

model_applications/medium_range/TCStat_SeriesAnalysis_fcstGFS _obsGFS_FeatureRelative _SeriesByInit.conf

Scientific Objective

By maintaining focus of each evaluation time (or evaluation time series, in this case) on a user-defined area around a cyclone, the model statistical errors associated with cyclonic physical features (moisture flux, stability, strength of upper-level PV anomaly and jet, etc.) can be related directly to the model forecasts and provide improvement guidance by accurately depicting interactions with significant weather features around and within the cyclone. This is in contrast to the traditional method of regional averaging cyclone observations in a fixed grid, which "smooths out" system features and limits the meaningful metrics that can be gathered.

Datasets

Relevant information about the datasets that would be beneficial include:

- TC-Pairs/TC-Stat Forecast dataset: ADeck modified-ATCF tropical cyclone data
- Series-Analysis Forecast dataset: GFS
- TC-Pairs/TC-Stat Observation dataset: BDeck modified-ATCF tropical cyclone data
- Series-Analysis Observation dataset: GFS Analysis

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* netCDF4
```

METplus Components

This use case first runs TCPairs and ExtractTiles to generate matched tropical cyclone data and regrid them into appropriately-sized tiles along a storm track. The MET tc-stat tool is used to filter the track data, and the MET regrid-dataplane tool is used to regrid the data (GRIB1 or GRIB2 into netCDF). Next, a series analysis by init time is performed on the results and plots (.ps and .png) are generated for all variable-level-stat combinations from the specified variables, levels, and requested statistics.

METplus Workflow

The following tools are used for each run time: TCPairs > RegridDataPlane, TCStat > SeriesAnalysis

This example loops by initialization time. For each initialization time it will process forecast leads 6, 12, 18, 24, 30, 36, and 40. There is only one initialization time in this example, so the following will be run:

Run times:

Init: 20141214_0Z
Forecast lead: 6

Init: 20141214_0Z
Forecast lead: 12

Init: 20141214_0Z
Forecast lead: 18

Init: 20141214_0Z
Forecast lead: 24

Init: 20141214_0Z

Forecast lead: 30

Init: 20141214_0Z
Forecast lead: 36

Init: 20141214_0Z
Forecast lead: 42

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesB

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/medium_range/TCStat_
↪SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByInit.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCPairs, TCStat, ExtractTiles, TCStat(for_series_analysis), SeriesAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
```

(continues on next page)

(continued from previous page)

```
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20141214
INIT_END = 20141214
INIT_INCREMENT = 21600

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = True


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/track_data
TC_PAIRS_ADECK_TEMPLATE = {date?fmt=%Y%m}/a{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}

TC_PAIRS_BDECK_INPUT_DIR = {TC_PAIRS_ADECK_INPUT_DIR}
TC_PAIRS_BDECK_TEMPLATE = {date?fmt=%Y%m}/b{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}

TC_PAIRS_REFORMAT_DIR = {OUTPUT_BASE}/track_data_atcf

TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs
TC_PAIRS_OUTPUT_TEMPLATE = {date?fmt=%Y%m}/{basin?fmt=%s}q{date?fmt=%Y%m%d%H}.gfso.{cyclone?
↪fmt=%s}

TC_PAIRS_SKIP_IF_REFORMAT_EXISTS = yes
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = yes


TC_STAT_LOOKIN_DIR = {TC_PAIRS_OUTPUT_DIR}

TC_STAT_OUTPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
TC_STAT_DUMP_ROW_TEMPLATE = {init?fmt=%Y%m%d_%H}/filter_{init?fmt=%Y%m%d_%H}.tcst


EXTRACT_TILES_TC_STAT_INPUT_DIR = {TC_STAT_OUTPUT_DIR}
```

(continues on next page)

(continued from previous page)

```
EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}

FCST_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/reduced_model_
↪data
FCST_EXTRACT_TILES_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}
↪00_{lead?fmt=%3H}.grb2

OBS_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/reduced_model_data
OBS_EXTRACT_TILES_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}
↪00_000.grb2

EXTRACT_TILES_OUTPUT_DIR = {OUTPUT_BASE}/extract_tiles
FCST_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/{storm_id}/FCST_TILE_F{lead?fmt=
↪%3H}_gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%3H}.nc
OBS_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/{storm_id}/OBS_TILE_F{lead?fmt=%3H}_
↪gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%3H}.nc

EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS = yes


SERIES_ANALYSIS_TC_STAT_INPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}
SERIES_ANALYSIS_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}

FCST_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = {FCST_EXTRACT_TILES_OUTPUT_TEMPLATE}

OBS_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = {OBS_EXTRACT_TILES_OUTPUT_TEMPLATE}

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/series_analysis_init
SERIES_ANALYSIS_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/{storm_id}/series_{fcst_name}_{fcst_
↪level}.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = GFSO

BOTH_VAR1_NAME = TMP
BOTH_VAR1_LEVELS = Z2
```

(continues on next page)

(continued from previous page)

```
###
# TCPairs Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcpairs
###

TC_PAIRS_INIT_INCLUDE =
TC_PAIRS_INIT_EXCLUDE =

TC_PAIRS_VALID_BEG =
TC_PAIRS_VALID_END =

TC_PAIRS_STORM_ID =
TC_PAIRS_BASIN =
TC_PAIRS_CYCLONE =
TC_PAIRS_STORM_NAME =

TC_PAIRS_DLAND_FILE = {MET_INSTALL_DIR}/share/met/tc_data/dland_global_tenth_degree.nc

TC_PAIRS_REFORMAT_DECK = yes
TC_PAIRS_REFORMAT_TYPE = SBU

TC_PAIRS_MISSING_VAL_TO_REPLACE = -99
TC_PAIRS_MISSING_VAL = -9999


###
# TCStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcstat
###

TC_STAT_JOB_ARGS = -job filter -basin ML -dump_row {TC_STAT_OUTPUT_DIR}/{TC_STAT_DUMP_ROW_
↪TEMPLATE}

TC_STAT_MATCH_POINTS = true

TC_STAT_AMODEL =
TC_STAT_BMODEL =
TC_STAT_DESC =
TC_STAT_STORM_ID =
TC_STAT_BASIN =
TC_STAT_CYCLONE =
TC_STAT_STORM_NAME =

TC_STAT_INIT_BEG =
TC_STAT_INIT_END =
```

(continues on next page)

(continued from previous page)

```
TC_STAT_INIT_INCLUDE = {init?fmt=%Y%m%d_%H}
TC_STAT_INIT_EXCLUDE =
TC_STAT_INIT_HOUR =

TC_STAT_VALID_BEG =
TC_STAT_VALID_END =
TC_STAT_VALID_INCLUDE =
TC_STAT_VALID_EXCLUDE =
TC_STAT_VALID_HOUR =
TC_STAT_LEAD_REQ =
TC_STAT_INIT_MASK =
TC_STAT_VALID_MASK =

TC_STAT_VALID_HOUR =
TC_STAT_LEAD =

TC_STAT_TRACK_WATCH_WARN =

TC_STAT_COLUMN_THRESH_NAME =
TC_STAT_COLUMN_THRESH_VAL =

TC_STAT_COLUMN_STR_NAME =
TC_STAT_COLUMN_STR_VAL =

TC_STAT_INIT_THRESH_NAME =
TC_STAT_INIT_THRESH_VAL =

TC_STAT_INIT_STR_NAME =
TC_STAT_INIT_STR_VAL =

TC_STAT_WATER_ONLY =

TC_STAT_LANDFALL =

TC_STAT_LANDFALL_BEG =
TC_STAT_LANDFALL_END =


###
# ExtractTiles Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#extracttiles
###

EXTRACT_TILES_NLAT = 60
EXTRACT_TILES_NLON = 60
```

(continues on next page)

(continued from previous page)

```
EXTRACT_TILES_DLAT = 0.5
EXTRACT_TILES_DLON = 0.5

EXTRACT_TILES_LON_ADJ = 15
EXTRACT_TILES_LAT_ADJ = 15


###
# TCStat (for SeriesAnalysis) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcstat
###

# Settings specific to the TCStat(for_series_analysis) process that was set
# in the PROCESS_LIST. Any TC_STAT_* variable not set in this section will use
# the value set outside of this section
[for_series_analysis]

TC_STAT_JOB_ARGS = -job filter -init_beg {INIT_BEG} -init_end {INIT_END} -dump_row {TC_STAT_
↪OUTPUT_DIR}/{TC_STAT_DUMP_ROW_TEMPLATE}

TC_STAT_OUTPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}
TC_STAT_LOOKIN_DIR = {EXTRACT_TILES_OUTPUT_DIR}


###
# SeriesAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#seriesanalysis
###

[config]

SERIES_ANALYSIS_BACKGROUND_MAP = no

SERIES_ANALYSIS_REGRID_TO_GRID = FCST
SERIES_ANALYSIS_REGRID_METHOD = FORCE

SERIES_ANALYSIS_STAT_LIST = TOTAL, FBAR, OBAR, ME

SERIES_ANALYSIS_BLOCK_SIZE = 4000

SERIES_ANALYSIS_IS_PAIRED = True

SERIES_ANALYSIS_GENERATE_PLOTS = yes
```

(continues on next page)

(continued from previous page)

```
SERIES_ANALYSIS_GENERATE_ANIMATIONS = no

PLOT_DATA_PLANE_TITLE = {MODEL} Init {init?fmt=%Y%m%d_%H} Storm {storm_id} {num_leads}␣
↪Forecasts (F{fcst_beg} to F{fcst_end}) {stat} for {fcst_name}, {fcst_level}
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

TCPairsConfig_wrapped

Note: See the *TCPairs MET Configuration* (page 282) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

//
// Models
//
${METPLUS_MODEL}

//
// Description
//
${METPLUS_DESC}

//
// Storm identifiers
```

(continues on next page)

(continued from previous page)

```
//
${METPLUS_STORM_ID}

//
// Basins
//
${METPLUS_BASIN}

//
// Cyclone numbers
//
${METPLUS_CYCLONE}

//
// Storm names
//
${METPLUS_STORM_NAME}

//
// Model initialization time windows to include or exclude
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
${METPLUS_VALID_INC}
// valid_exc =
${METPLUS_VALID_EXC}

// write_valid =
${METPLUS_WRITE_VALID}

//
// Valid model time window
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}

//
// Model initialization hours
//
```

(continues on next page)

(continued from previous page)

```
init_hour = [];

//
// Required lead time in hours
//
lead_req = [];

//
// lat/lon polylines defining masking regions
//
init_mask  = "";
valid_mask = "";

//
// Specify if the code should check for duplicate ATCF lines
//
//check_dup =
${METPLUS_CHECK_DUP}


//
// Specify special processing to be performed for interpolated models.
// Set to NONE, FILL, or REPLACE.
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
//
//consensus =
${METPLUS_CONSENSUS_LIST}


//
// Forecast lag times
//
lag_time = [];

//
// CLIPER/SHIFOR baseline forecasts to be derived from the BEST
// and operational (CARQ) tracks.
//
best_technique = [ "BEST" ];
best_baseline  = [];
```

(continues on next page)

(continued from previous page)

```
oper_technique = [ "CARQ" ];
oper_baseline  = [];

//
// Specify the datasets to be searched for analysis tracks (NONE, ADECK, BDECK,
// or BOTH).
//
anly_track = BDECK;

//
// Specify if only those track points common to both the ADECK and BDECK
// tracks be written out.
//
//match_points =
${METPLUS_MATCH_POINTS}

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
${METPLUS_DLAND_FILE}

//
// Specify watch/warning information:
//   - Input watch/warning filename
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}


//diag_info_map = {
${METPLUS_DIAG_INFO_MAP_LIST}

//diag_convert_map = {
${METPLUS_DIAG_CONVERT_MAP_LIST}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V9.0";
```

(continues on next page)

(continued from previous page)

```
tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

TCStatConfig_wrapped

Note: See the *TCStat MET Configuration* (page 297) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCStat configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to filter the TC-STAT data down to the
// desired subset of lines over which statistics are to be computed.  Only
// those lines which meet ALL of the criteria specified will be retained.
//
// The settings that are common to all jobs may be specified once at the top
// level.  If no selection is listed for a parameter, that parameter will not
// be used for filtering.  If multiple selections are listed for a parameter,
// the analyses will be performed on their union.
//

//
// Stratify by the AMODEL or BMODEL columns.
//
${METPLUS_AMODEL}
${METPLUS_BMODEL}

//
// Stratify by the DESC column.
//
${METPLUS_DESC}

//
// Stratify by the STORM_ID column.
//
${METPLUS_STORM_ID}

//
// Stratify by the BASIN column.
```

(continues on next page)

(continued from previous page)

```
// May add using the "-basin" job command option.
//
${METPLUS_BASIN}

//
// Stratify by the CYCLONE column.
// May add using the "-cyclone" job command option.
//
${METPLUS_CYCLONE}

//
// Stratify by the STORM_NAME column.
// May add using the "-storm_name" job command option.
//
${METPLUS_STORM_NAME}

//
// Stratify by the INIT times.
// Model initialization time windows to include or exclude
// May modify using the "-init_beg", "-init_end", "-init_inc",
// and "-init_exc" job command options.
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
${METPLUS_INIT_INC}
${METPLUS_INIT_EXC}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INC}
${METPLUS_VALID_EXC}

//
// Stratify by the initialization and valid hours and lead time.
//
${METPLUS_INIT_HOUR}
${METPLUS_VALID_HOUR}
${METPLUS_LEAD}

//
// Select tracks which contain all required lead times.
//
```

(continues on next page)

(continued from previous page)

```
${METPLUS_LEAD_REQ}

//
// Stratify by the INIT_MASK and VALID_MASK columns.
//
${METPLUS_INIT_MASK}
${METPLUS_VALID_MASK}


//
// Stratify by the LINE_TYPE column.
//
//line_type =
${METPLUS_LINE_TYPE}


//
// Stratify by checking the watch/warning status for each track point
// common to both the ADECK and BDECK tracks.  If the watch/warning status
// of any of the track points appears in the list, retain the entire track.
//
${METPLUS_TRACK_WATCH_WARN}

//
// Stratify by applying thresholds to numeric data columns.
//
${METPLUS_COLUMN_THRESH_NAME}
${METPLUS_COLUMN_THRESH_VAL}

//
// Stratify by performing string matching on non-numeric data columns.
//
${METPLUS_COLUMN_STR_NAME}
${METPLUS_COLUMN_STR_VAL}

//
// Stratify by excluding strings in non-numeric data columns.
//
//column_str_exc_name =
${METPLUS_COLUMN_STR_EXC_NAME}

//column_str_exc_val =
${METPLUS_COLUMN_STR_EXC_VAL}

//
```

(continues on next page)

(continued from previous page)

```
// Similar to the column_thresh options above
//
${METPLUS_INIT_THRESH_NAME}
${METPLUS_INIT_THRESH_VAL}

//
// Similar to the column_str options above
//
${METPLUS_INIT_STR_NAME}
${METPLUS_INIT_STR_VAL}

//
// Similar to the column_str_exc options above
//
//init_str_exc_name =
${METPLUS_INIT_STR_EXC_NAME}

//init_str_exc_val =
${METPLUS_INIT_STR_EXC_VAL}

//diag_thresh_name =
${METPLUS_DIAG_THRESH_NAME}

//diag_thresh_val =
${METPLUS_DIAG_THRESH_VAL}

//init_diag_thresh_name =
${METPLUS_INIT_DIAG_THRESH_NAME}

//init_diag_thresh_val =
${METPLUS_INIT_DIAG_THRESH_VAL}


//
// Stratify by the ADECK and BDECK distances to land.
//
${METPLUS_WATER_ONLY}

//
// Specify whether only those track points occurring near landfall should be
// retained, and define the landfall retention window in HH[MMSS] format
// around the landfall time.
//
${METPLUS_LANDFALL}
${METPLUS_LANDFALL_BEG}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_LANDFALL_END}

//
// Specify whether only those track points common to both the ADECK and BDECK
// tracks should be retained.  May modify using the "-match_points" job command
// option.
//
${METPLUS_MATCH_POINTS}

//event_equal =
${METPLUS_EVENT_EQUAL}

//event_equal_lead =
${METPLUS_EVENT_EQUAL_LEAD}

//out_init_mask =
${METPLUS_OUT_INIT_MASK}

//out_valid_mask =
${METPLUS_OUT_VALID_MASK}


//
// Array of TCStat analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

SeriesAnalysisConfig_wrapped

Note: See the *SeriesAnalysis MET Configuration* (page 232) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Series-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
//
//desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
//obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

censor_thresh = [];
censor_val    = [];
//cat_thresh =
${METPLUS_CAT_THRESH}
cnt_thresh    = [ NA ];
cnt_logic     = UNION;

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_CAT_THRESH}
   ${METPLUS_FCST_FIELD}
}
obs = {
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

//
// Number of grid points to be processed concurrently.  Set smaller to use
```

(continues on next page)

(continued from previous page)

```
// less memory but increase the number of passes through the data.
//
//block_size =
${METPLUS_BLOCK_SIZE}

//
// Ratio of valid matched pairs to compute statistics for a grid point
//
//vld_thresh =
${METPLUS_VLD_THRESH}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_stats = {
${METPLUS_OUTPUT_STATS_DICT}

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByInit.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByInit.conf -c /path/to/
↪user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByInit.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByInit.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

  and for the [exe] section, you will need to define the location of NON-MET executables. If the executable is in the user's path, METplus will find it from the name. If the executable is not in the path, specify the full path to the executable here (i.e. CONVERT = /usr/bin/convert) The following executables are required for performing series analysis use cases:

  If the executables are in the path:

- **CONVERT = convert**

NOTE: All of these executable items must be located under the [exe] section.

If the executables are not in the path, they need to be defined:

- **CONVERT = /path/to/convert**

NOTE: All of these executable items must be located under the [exe] section. Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y

[exe]
CONVERT = /path/to/convert
```

NOTE: The INPUT_BASE, OUTPUT_BASE, and MET_INSTALL_DIR must be located under the [dir] section, while the RM, CUT, TR, NCAP2, CONVERT, and NCDUMP must be located under the [exe] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in series_analysis_init/20141214_00 (relative to **OUTPUT_BASE**) and will contain the following subdirectories:

- ML1200942014
- ML1200942014
- ML1200942014
- ML1201002014
- ML1201032014
- ML1201042014
- ML1201052014
- ML1201062014
- ML1201072014
- ML1201082014
- ML1201092014
- ML1201102014

Each subdirectory will contain files that have the following format:

ANLY_ASCII_FILES_<storm>

FCST_ASCII_FILES_<storm>

series_<varname>_<level>_<stat>.png

series_<varname>_<level>_<stat>.ps

series_<varname>_<level>_<stat>.nc

Keywords

Note:

- TCStatToolUseCase
- SeriesByInitUseCase
- RegridDataPlaneToolUseCase
- MediumRangeAppUseCase

- SeriesAnalysisUseCase
- GRIB2FileUseCase
- TCPairsToolUseCase
- FeatureRelativeUseCase
- SBUOrgUseCase
- DiagnosticsUseCase
- RuntimeFreqUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/medium_range-TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_Se

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.7.5 Multi_Tool: Feature Relative by Lead using Multiple User-Defined Fields

model_applications/medium_range/ TCStat_SeriesAnalysis_fcstGFS _obsGFS_FeatureRelative _SeriesByLead_PyEmbed_Multiple_Diagnostics.conf

Scientific Objective

This use case calls multiple tools to produce diagnostic plots of systematic erros relative to a feature (e.g. hurricane, MCS, etc...). This use case calls two user provided python scripts that calculate diagnostics of interest (e.g. integrated vapor transport, potential vorticity, etc...). These user diagnostics are then used to define the systematic errors. This example calculates statistics over varying forecast leads with the ability to define lead groupings. This use case is very similar to the Multi_Tools: Feature Relative by Lead use case and the Multi_Tools: Feature Relative by Lead using User-Defined Fields. (ADeck,GFS:BDeck,GFS:ATCF,Grib2)

By maintaining focus of each evaluation time (or evaluation time series, in this case) on a user-defined area around a cyclone, the model statistical errors associated with cyclonic physical features (moisture flux, stability, strength of upper-level PV anomaly and jet, etc.) can be related directly to the model forecasts and provide improvement guidance by accurately depicting interactions with significant weather features around and within the cyclone. This is in contrast to the traditional method of regional averaging cyclone observations in a fixed grid, which "smooths out" system features and limits the meaningful metrics that can be gathered. Specifically, this use case creates bins of forecast lead times as specified by the given ranges which provides additional insight directly into forecast lead time accuracy.

Additionally, the ability to calculate model statistical errors based on user provided diagnostics allows the user to customize the feature relative analysis to suit their needs.

Datasets

This use case compares the Global Forecast System (GFS) forecast to the GFS analysis for hurricane Dorian. It is based on three user provided python scripts that calculate the diagnostic integrated vaport transport (IVT) baroclinic potential vorticity (PV), and saturation equivalent potential temperature (SEPT), respectively.

- Variables required to calculate IVT: Levels required: all pressure levels >= 100mb #. Temperature #. v- component of wind #. u- component of wind #. Geopotential height #. Specific humidity OR Relative Humidity
- Variables required to calculate PV: Levels required: all pressure levels >= 100mb #. U-wind #. V-wind #. Temperature
- Variables required to calculate saturation equivalent potential temperature: Levels required: all pressure levels >= 100mb #. Temperature
- Forecast dataset: GFS Grid 4 Forecast GFS Forecast data can be found at the following website: https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs - Initialization date: 20190830 - Initialization hours: 00, 06, 12, 18 UTC - Lead times: 90, 96, 102, 108, 114 - Format: Grib2 - Resolution: 0.5 degree
- Observation dataset: GFS Grid 4 Analysis GFS Analysis data can be found at the following website: https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs - Valid date/time range: 20190902_18 - 20190904_12 every 6 hours - Format: Grib2 - Resolution: 0.5 degree
- Hurricane Track Data Hurricane track data can be found at the following website: http://hurricanes.ral.ucar.edu/repository/data/ - ADeck Track File: aal052019.dat - BDeck Track File: bal052019.dat

External Dependencies

You will need to use a version of Python 3.7+ that has the following packages installed:

- netCDF4
- pygrib
- cfgrib
- metpy
- xarray

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

```
[user_env_vars]
MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python
```

METplus Components

This use case first runs PyEmbedIngest to run the user provided python scripts to calculate the desired diagnostics (in this example, IVT, PV and SEPT). PyEmbedIngest runs the RegridDataPlane tool to write IVT, PV, and SEPTto a MET readable netCDF file. Then TCPairs and ExtractTiles are run to generate matched tropical cyclone data and regrid them into appropriately-sized tiles along a storm track. The MET tc-stat tool is used to filter the track data and the MET regrid-dataplane tool is used to regrid the data (GRIB1 or GRIB2 into netCDF). Next, a series analysis by lead time is performed on the results and plots (.ps and .png) are generated for all variable-level-stat combinations from the specified variables, levels, and requested statistics. If lead grouping is turned on, the final results are aggregated into forecast hour groupings as specified by the start, end and increment in the METplus configuration file, as well as labels to identify each forecast hour grouping. If lead grouping is not turned out the final results will be written out for each requested lead time.

METplus Workflow

This use case loops by process which means that each tool is run for all times before moving to the next tool. The tool order is as follows:

PyEmbedIngest, TCPairs, ExtractTiles, SeriesByLead

This example loops by forecast/lead time (with begin, end, and increment as specified in the METplus TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead_Multiple_Diagnostics.conf file).

4 initialization times will be run over 5 lead times:

Init: 20190830_00Z
Forecast lead: 90, 96, 102, 108, 114

Init: 20190830_06Z
Forecast lead: 90, 96, 102, 108, 114

Init: 20190830_12Z
Forecast lead: 90, 96, 102, 108, 114

Init: 20190830_18Z
Forecast lead: 90, 96, 102, 108, 114

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesB

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/medium_range/TCStat_
↪SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead_PyEmbed_Multiple_Diagnostics.
↪html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PyEmbedIngest, TCPairs, TCStat, ExtractTiles, TCStat(for_series_analysis),␣
↪SeriesAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2019083000
INIT_END = 2019083023
INIT_INCREMENT = 21600

LEAD_SEQ = 90, 96, 102, 108, 114

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_LEAD
```

(continues on next page)

(continued from previous page)

```
SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

PY_EMBED_INGEST_1_OUTPUT_DIR = {OUTPUT_BASE}/py_embed_out
PY_EMBED_INGEST_1_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d}/gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}
↪00_{lead?fmt=%3H}.nc

PY_EMBED_INGEST_2_OUTPUT_DIR = {PY_EMBED_INGEST_1_OUTPUT_DIR}
PY_EMBED_INGEST_2_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=
↪%H}00_000.nc


TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/dorian_data/track_
↪data
TC_PAIRS_ADECK_TEMPLATE = a{basin?fmt=%s}052019.dat

TC_PAIRS_BDECK_INPUT_DIR = {TC_PAIRS_ADECK_INPUT_DIR}
TC_PAIRS_BDECK_TEMPLATE = b{basin?fmt=%s}052019.dat

TC_PAIRS_REFORMAT_DIR = {OUTPUT_BASE}/track_data_atcf
TC_PAIRS_SKIP_IF_REFORMAT_EXISTS = no

TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs
TC_PAIRS_OUTPUT_TEMPLATE = {date?fmt=%Y%m}/{basin?fmt=%s}q{date?fmt=%Y%m%d%H}.dorian
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = no


TC_STAT_LOOKIN_DIR = {TC_PAIRS_OUTPUT_DIR}

TC_STAT_OUTPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
TC_STAT_DUMP_ROW_TEMPLATE = filter_{init?fmt=%Y%m%d_%H}.tcst


EXTRACT_TILES_TC_STAT_INPUT_DIR = {TC_STAT_OUTPUT_DIR}
EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}

EXTRACT_TILES_GRID_INPUT_DIR = {PY_EMBED_INGEST_1_OUTPUT_DIR}
FCST_EXTRACT_TILES_INPUT_DIR = {PY_EMBED_INGEST_1_OUTPUT_DIR}
FCST_EXTRACT_TILES_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}
```

(continues on next page)

(continued from previous page)

```
↪00_{lead?fmt=%3H}.nc

OBS_EXTRACT_TILES_INPUT_DIR = {PY_EMBED_INGEST_1_OUTPUT_DIR}
OBS_EXTRACT_TILES_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}
↪00_000.nc

EXTRACT_TILES_OUTPUT_DIR = {OUTPUT_BASE}/extract_tiles
FCST_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/{storm_id}/FCST_TILE_F{lead?fmt=
↪%3H}_{MODEL}_gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%3H}.nc
OBS_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/{storm_id}/OBS_TILE_F{lead?fmt=%3H}_
↪{MODEL}_gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%3H}.nc
EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS = no


FCST_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = {FCST_EXTRACT_TILES_OUTPUT_TEMPLATE}

OBS_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = {OBS_EXTRACT_TILES_OUTPUT_TEMPLATE}

SERIES_ANALYSIS_TC_STAT_INPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}
SERIES_ANALYSIS_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/series_analysis_lead
SERIES_ANALYSIS_OUTPUT_TEMPLATE = {label}/series_F{fcst_beg}_{fcst_name}_{fcst_level}.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

# Used by extract tiles and series analysis to define the records of
#  interest to be retrieved from the grib2 file

MODEL = GFSO

BOTH_VAR1_NAME = ivt
BOTH_VAR1_LEVELS = Surface

BOTH_VAR2_NAME = pv
BOTH_VAR2_LEVELS = Surface

BOTH_VAR3_NAME = sept
BOTH_VAR3_LEVELS = Surface
```

(continues on next page)

(continued from previous page)

```
###
# PyEmbedIngest Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pyembedingest
###

CONFIG_DIR={PARM_BASE}/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_
↪fcstGFS_obsGFS_FeatureRelative_SeriesByLead_PyEmbed_Multiple_Diagnostics
MODEL_DIR = {INPUT_BASE}/model_applications/medium_range/dorian_data/model_data

# 1st INGEST INSTANCE: Forecast

# IVT
PY_EMBED_INGEST_1_SCRIPT_1 = {CONFIG_DIR}/gfs_ivt_fcst.py {MODEL_DIR}/{init?fmt=%Y%m%d}/gfs_
↪4_{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%3H}.grb2
PY_EMBED_INGEST_1_OUTPUT_FIELD_NAME_1 = ivt

# PV
PY_EMBED_INGEST_1_SCRIPT_2 = {CONFIG_DIR}/gfs_pv_fcst.py {MODEL_DIR}/{init?fmt=%Y%m%d}/gfs_4_
↪{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%3H}.grb2
PY_EMBED_INGEST_1_OUTPUT_FIELD_NAME_2 = pv

# SEPT
PY_EMBED_INGEST_1_SCRIPT_3 = {CONFIG_DIR}/gfs_sept_fcst.py {MODEL_DIR}/{init?fmt=%Y%m%d}/gfs_
↪4_{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%3H}.grb2
PY_EMBED_INGEST_1_OUTPUT_FIELD_NAME_3 = sept

PY_EMBED_INGEST_1_TYPE = NUMPY

PY_EMBED_INGEST_1_OUTPUT_GRID = {MODEL_DIR}/{init?fmt=%Y%m%d}/gfs_4_{init?fmt=%Y%m%d}_{init?
↪fmt=%H}00_{lead?fmt=%3H}.grb2

# 2nd INGEST INSTANCE: Analysis

# IVT
PY_EMBED_INGEST_2_SCRIPT_1 = {CONFIG_DIR}/gfs_ivt_analysis.py {MODEL_DIR}/{valid?fmt=%Y%m%d}/
↪gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}00_000.grb2
PY_EMBED_INGEST_2_OUTPUT_FIELD_NAME_1 = ivt

# PV
PY_EMBED_INGEST_2_SCRIPT_2 = {CONFIG_DIR}/gfs_pv_analysis.py {MODEL_DIR}/{valid?fmt=%Y%m%d}/
↪gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}00_000.grb2
PY_EMBED_INGEST_2_OUTPUT_FIELD_NAME_2 = pv
```

(continues on next page)

(continued from previous page)

```
# SEPT
PY_EMBED_INGEST_2_SCRIPT_3 = {CONFIG_DIR}/gfs_sept_analysis.py {MODEL_DIR}/{valid?fmt=%Y%m%d}
↪/gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}00_000.grb2
PY_EMBED_INGEST_2_OUTPUT_FIELD_NAME_3 = sept

PY_EMBED_INGEST_2_TYPE = NUMPY

PY_EMBED_INGEST_2_OUTPUT_GRID = {MODEL_DIR}/{valid?fmt=%Y%m%d}/gfs_4_{valid?fmt=%Y%m%d}_
↪{valid?fmt=%H}00_000.grb2


###
# TCPairs Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcpairs
###

TC_PAIRS_SKIP_LEAD_SEQ = True

TC_PAIRS_INIT_INCLUDE =
TC_PAIRS_INIT_EXCLUDE =

TC_PAIRS_VALID_BEG =
TC_PAIRS_VALID_END =

TC_PAIRS_STORM_ID =
TC_PAIRS_BASIN =
TC_PAIRS_CYCLONE =
TC_PAIRS_STORM_NAME =

TC_PAIRS_DLAND_FILE = {MET_INSTALL_DIR}/share/met/tc_data/dland_global_tenth_degree.nc

TC_PAIRS_REFORMAT_DECK = no
TC_PAIRS_REFORMAT_TYPE = SBU

TC_PAIRS_MISSING_VAL_TO_REPLACE = -99
TC_PAIRS_MISSING_VAL = -9999


###
# TCStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcstat
###

TC_STAT_JOB_ARGS = -job filter -basin AL -dump_row {TC_STAT_OUTPUT_DIR}/{TC_STAT_DUMP_ROW_
↪TEMPLATE}
```

(continues on next page)

(continued from previous page)

```
TC_STAT_MATCH_POINTS = true

TC_STAT_AMODEL = {MODEL}
TC_STAT_BMODEL =
TC_STAT_DESC =
TC_STAT_STORM_ID =
TC_STAT_BASIN =
TC_STAT_CYCLONE =
TC_STAT_STORM_NAME =

TC_STAT_INIT_BEG =
TC_STAT_INIT_END =
TC_STAT_INIT_INCLUDE = {init?fmt=%Y%m%d_%H}
TC_STAT_INIT_EXCLUDE =
TC_STAT_INIT_HOUR =

TC_STAT_VALID_BEG =
TC_STAT_VALID_END =
TC_STAT_VALID_INCLUDE =
TC_STAT_VALID_EXCLUDE =
TC_STAT_VALID_HOUR =
TC_STAT_LEAD_REQ =
TC_STAT_INIT_MASK =
TC_STAT_VALID_MASK =

TC_STAT_VALID_HOUR =
TC_STAT_LEAD =

TC_STAT_TRACK_WATCH_WARN =

TC_STAT_COLUMN_THRESH_NAME =
TC_STAT_COLUMN_THRESH_VAL =

TC_STAT_COLUMN_STR_NAME =
TC_STAT_COLUMN_STR_VAL =

TC_STAT_INIT_THRESH_NAME =
TC_STAT_INIT_THRESH_VAL =

TC_STAT_INIT_STR_NAME =
TC_STAT_INIT_STR_VAL =

TC_STAT_WATER_ONLY =
```

(continues on next page)

(continued from previous page)

```
TC_STAT_LANDFALL =

TC_STAT_LANDFALL_BEG =
TC_STAT_LANDFALL_END =


###
# ExtractTiles Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#extracttiles
###

EXTRACT_TILES_NLAT = 60
EXTRACT_TILES_NLON = 60

EXTRACT_TILES_DLAT = 0.5
EXTRACT_TILES_DLON = 0.5

EXTRACT_TILES_LON_ADJ = 15
EXTRACT_TILES_LAT_ADJ = 15


###
# TCStat (for SeriesAnalysis) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcstat
###

# Settings specific to the TCStat(for_series_analysis) process that was set
# in the PROCESS_LIST. Any TC_STAT_* variable not set in this section will use
# the value set outside of this section

[for_series_analysis]

TC_STAT_JOB_ARGS = -job filter -init_beg {INIT_BEG} -init_end {INIT_END} -dump_row {TC_STAT_
↪OUTPUT_DIR}/{TC_STAT_DUMP_ROW_TEMPLATE}

TC_STAT_OUTPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}
TC_STAT_LOOKIN_DIR = {EXTRACT_TILES_OUTPUT_DIR}


###
# SeriesAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#seriesanalysis
###

[config]
```

(continues on next page)

(continued from previous page)

```
SERIES_ANALYSIS_REGRID_TO_GRID = FCST
SERIES_ANALYSIS_REGRID_METHOD = FORCE

SERIES_ANALYSIS_STAT_LIST = TOTAL, FBAR, OBAR, ME

SERIES_ANALYSIS_BACKGROUND_MAP = yes

SERIES_ANALYSIS_BLOCK_SIZE = 4000

SERIES_ANALYSIS_IS_PAIRED = True

SERIES_ANALYSIS_GENERATE_PLOTS = yes
SERIES_ANALYSIS_GENERATE_ANIMATIONS = yes

PLOT_DATA_PLANE_TITLE = {MODEL} series_F{fcst_beg} Forecasts{nseries}, {stat} for {fcst_name}
↪ {fcst_level}


[user_env_vars]

PV_LAYER_MIN_PRESSURE=100.0
PV_LAYER_MAX_PRESSURE=1000.0
IVT_LAYER_MIN_PRESSURE=100.0
IVT_LAYER_MAX_PRESSURE=1000.0
SEPT_LAYER_MIN_PRESSURE=100.0
SEPT_LAYER_MAX_PRESSURE=1000.0
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

TCPairsConfig_wrapped

Note: See the *TCPairs MET Configuration* (page 282) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

//
// Models
//
${METPLUS_MODEL}

//
// Description
//
${METPLUS_DESC}

//
// Storm identifiers
//
${METPLUS_STORM_ID}

//
// Basins
//
${METPLUS_BASIN}

//
// Cyclone numbers
//
${METPLUS_CYCLONE}

//
// Storm names
//
${METPLUS_STORM_NAME}

//
// Model initialization time windows to include or exclude
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
```

(continues on next page)

(continued from previous page)

```
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
${METPLUS_VALID_INC}
// valid_exc =
${METPLUS_VALID_EXC}

// write_valid =
${METPLUS_WRITE_VALID}

//
// Valid model time window
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}

//
// Model initialization hours
//
init_hour = [];

//
// Required lead time in hours
//
lead_req = [];

//
// lat/lon polylines defining masking regions
//
init_mask  = "";
valid_mask = "";

//
// Specify if the code should check for duplicate ATCF lines
//
//check_dup =
${METPLUS_CHECK_DUP}


//
// Specify special processing to be performed for interpolated models.
// Set to NONE, FILL, or REPLACE.
```

(continues on next page)

(continued from previous page)

```
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
//
//consensus =
${METPLUS_CONSENSUS_LIST}


//
// Forecast lag times
//
lag_time = [];

//
// CLIPER/SHIFOR baseline forecasts to be derived from the BEST
// and operational (CARQ) tracks.
//
best_technique = [ "BEST" ];
best_baseline  = [];
oper_technique = [ "CARQ" ];
oper_baseline  = [];

//
// Specify the datasets to be searched for analysis tracks (NONE, ADECK, BDECK,
// or BOTH).
//
anly_track = BDECK;

//
// Specify if only those track points common to both the ADECK and BDECK
// tracks be written out.
//
//match_points =
${METPLUS_MATCH_POINTS}

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
${METPLUS_DLAND_FILE}

//
```

(continues on next page)

(continued from previous page)

```
// Specify watch/warning information:
//   - Input watch/warning filename
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}


//diag_info_map = {
${METPLUS_DIAG_INFO_MAP_LIST}

//diag_convert_map = {
${METPLUS_DIAG_CONVERT_MAP_LIST}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

TCStatConfig_wrapped

Note: See the *TCStat MET Configuration* (page 297) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCStat configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to filter the TC-STAT data down to the
// desired subset of lines over which statistics are to be computed.  Only
// those lines which meet ALL of the criteria specified will be retained.
//
// The settings that are common to all jobs may be specified once at the top
// level.  If no selection is listed for a parameter, that parameter will not
```

(continues on next page)

(continued from previous page)

```
// be used for filtering.  If multiple selections are listed for a parameter,
// the analyses will be performed on their union.
//

//
// Stratify by the AMODEL or BMODEL columns.
//
${METPLUS_AMODEL}
${METPLUS_BMODEL}

//
// Stratify by the DESC column.
//
${METPLUS_DESC}

//
// Stratify by the STORM_ID column.
//
${METPLUS_STORM_ID}

//
// Stratify by the BASIN column.
// May add using the "-basin" job command option.
//
${METPLUS_BASIN}

//
// Stratify by the CYCLONE column.
// May add using the "-cyclone" job command option.
//
${METPLUS_CYCLONE}

//
// Stratify by the STORM_NAME column.
// May add using the "-storm_name" job command option.
//
${METPLUS_STORM_NAME}

//
// Stratify by the INIT times.
// Model initialization time windows to include or exclude
// May modify using the "-init_beg", "-init_end", "-init_inc",
// and "-init_exc" job command options.
//
${METPLUS_INIT_BEG}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_INIT_END}
${METPLUS_INIT_INC}
${METPLUS_INIT_EXC}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INC}
${METPLUS_VALID_EXC}

//
// Stratify by the initialization and valid hours and lead time.
//
${METPLUS_INIT_HOUR}
${METPLUS_VALID_HOUR}
${METPLUS_LEAD}

//
// Select tracks which contain all required lead times.
//
${METPLUS_LEAD_REQ}

//
// Stratify by the INIT_MASK and VALID_MASK columns.
//
${METPLUS_INIT_MASK}
${METPLUS_VALID_MASK}


//
// Stratify by the LINE_TYPE column.
//
//line_type =
${METPLUS_LINE_TYPE}


//
// Stratify by checking the watch/warning status for each track point
// common to both the ADECK and BDECK tracks.  If the watch/warning status
// of any of the track points appears in the list, retain the entire track.
//
${METPLUS_TRACK_WATCH_WARN}
```

(continues on next page)

(continued from previous page)

```
//
// Stratify by applying thresholds to numeric data columns.
//
${METPLUS_COLUMN_THRESH_NAME}
${METPLUS_COLUMN_THRESH_VAL}

//
// Stratify by performing string matching on non-numeric data columns.
//
${METPLUS_COLUMN_STR_NAME}
${METPLUS_COLUMN_STR_VAL}

//
// Stratify by excluding strings in non-numeric data columns.
//
//column_str_exc_name =
${METPLUS_COLUMN_STR_EXC_NAME}

//column_str_exc_val =
${METPLUS_COLUMN_STR_EXC_VAL}

//
// Similar to the column_thresh options above
//
${METPLUS_INIT_THRESH_NAME}
${METPLUS_INIT_THRESH_VAL}

//
// Similar to the column_str options above
//
${METPLUS_INIT_STR_NAME}
${METPLUS_INIT_STR_VAL}

//
// Similar to the column_str_exc options above
//
//init_str_exc_name =
${METPLUS_INIT_STR_EXC_NAME}

//init_str_exc_val =
${METPLUS_INIT_STR_EXC_VAL}

//diag_thresh_name =
${METPLUS_DIAG_THRESH_NAME}
```

(continues on next page)

(continued from previous page)

```
//diag_thresh_val =
${METPLUS_DIAG_THRESH_VAL}

//init_diag_thresh_name =
${METPLUS_INIT_DIAG_THRESH_NAME}

//init_diag_thresh_val =
${METPLUS_INIT_DIAG_THRESH_VAL}


//
// Stratify by the ADECK and BDECK distances to land.
//
${METPLUS_WATER_ONLY}

//
// Specify whether only those track points occurring near landfall should be
// retained, and define the landfall retention window in HH[MMSS] format
// around the landfall time.
//
${METPLUS_LANDFALL}
${METPLUS_LANDFALL_BEG}
${METPLUS_LANDFALL_END}

//
// Specify whether only those track points common to both the ADECK and BDECK
// tracks should be retained.  May modify using the "-match_points" job command
// option.
//
${METPLUS_MATCH_POINTS}

//event_equal =
${METPLUS_EVENT_EQUAL}

//event_equal_lead =
${METPLUS_EVENT_EQUAL_LEAD}

//out_init_mask =
${METPLUS_OUT_INIT_MASK}

//out_valid_mask =
${METPLUS_OUT_VALID_MASK}


//
```

(continues on next page)

(continued from previous page)

```
// Array of TCStat analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

SeriesAnalysisConfig_wrapped

Note: See the *SeriesAnalysis MET Configuration* (page 232) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Series-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
//
//desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
//obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
```

(continues on next page)

(continued from previous page)

```
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

censor_thresh = [];
censor_val    = [];
//cat_thresh =
${METPLUS_CAT_THRESH}
cnt_thresh    = [ NA ];
cnt_logic     = UNION;

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_CAT_THRESH}
   ${METPLUS_FCST_FIELD}
}
obs = {
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

//
// Number of grid points to be processed concurrently.  Set smaller to use
// less memory but increase the number of passes through the data.
//
//block_size =
${METPLUS_BLOCK_SIZE}

//
// Ratio of valid matched pairs to compute statistics for a grid point
//
//vld_thresh =
${METPLUS_VLD_THRESH}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_stats = {
${METPLUS_OUTPUT_STATS_DICT}

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
```

(continues on next page)

(continued from previous page)

```
rank_corr_flag = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses four Python embedding scripts to read input data, two for the forecast data and two for the analysis data. The multiple datatype input requires the two-script approach.

parm/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesB

```
# This script is a combination of two scripts originally from Taylor Mandelbaum, SBU.
# Adjustments have been made by Lindsay Blank, NCAR.
# May 2020
################################################################################################
↪######

import pygrib
import numpy as np
import sys
import os
import re
import datetime as dt
import metpy.calc as mc

################################################################################################
↪######

def ivt(input_file):
    grbs = pygrib.open(input_file)
    g = 9.81    # Setting gravity constant
    print(input_file)
    grbs.rewind()

    # Initialize variable arrays
    levs = [] # Levels
    q    = [] # Specific humidity
    hgt  = [] # Geopotential height
    temp = [] # Temperature
```

(continues on next page)

(continued from previous page)

```
    u    = [] # u-wind
    v    = [] # v-wind

    # First obtain the levels we will use
    # These are in hPa in the file, so directly compare with user supplied min/max
    levs = sorted(set([grb.level for grb in grbs if float(grb.level) >= float(os.environ.get(
↪'IVT_LAYER_MIN_PRESSURE',100.0)) and float(grb.level) <= float(os.environ.get('IVT_LAYER_
↪MAX_PRESSURE',1000.0))]))

    # Fill in variable arrays from input file.
    grbs.rewind()
    for grb in grbs:
        if not grb.level in levs:
            continue
        elif np.logical_and('v-' in grb.parameterName,grb.typeOfLevel=='isobaricInhPa'):
            v.append(grb.values)
        elif np.logical_and('u-' in grb.parameterName,grb.typeOfLevel=='isobaricInhPa'):
            u.append(grb.values)
        elif np.logical_and('Temperature' in grb.parameterName,grb.typeOfLevel==
↪'isobaricInhPa'):
            temp.append(grb.values)
        elif np.logical_and('Geopotential' in grb.parameterName,grb.typeOfLevel==
↪'isobaricInhPa'):
            hgt.append(grb.values)
        elif np.logical_and('Specific' in grb.parameterName,grb.typeOfLevel=='isobaricInhPa
↪'):
            q.append(grb.values)

    temp = np.array(temp)
    hgt  = np.array(hgt)
    u    = np.array(u)
    v    = np.array(v)

    grbs.rewind()

    # If we didn't find specific humidity, look for relative humidity.
    if len(q) == 0:
        for grb in grbs:
            if not grb.level in levs:
                continue
            if np.logical_and('Relative' in grb.parameterName,grb.typeOfLevel=='isobaricInhPa
↪'):
                q.append(grb.values)

        levs = np.array(levs)
```

(continues on next page)

(continued from previous page)

```
        # Clausius-Clapeyron time
        es = 610.78*np.exp((17.67*(temp-273.15)/(temp-29.65))) # Calculate saturation vapor␣
↪pressure
        e = es*(np.array(q)/100) # Calculate vapor pressure
        w = 0.622*es/(levs[:,None,None]*100) # Calculate water vapor
        q = w/(w+1) # Calculate specific humidity
    q = np.array(q)

    uv = np.sqrt(u**2+v**2) # Calculate wind
    mflux_total = np.sum(q,axis=0)*(1/g)*np.mean(uv,axis=0)*(np.max(levs)-np.min(levs))
↪#calculate mass flux
    met_data = mflux_total.copy() #Pass mass flux to be used by MET tools
    print(np.max(met_data))
    #np.save('{}.npy'.format(sys.argv[1]),mflux_total)
    grbs.close()

    return met_data

################################################################################################
↪######

input_file = os.path.expandvars(sys.argv[1])

data = ivt(input_file) #Call function to calculate IVT

met_data = data
met_data = met_data.astype('float64')

# Automatically fill out time information from input file.
file_regex = r"^.*([0-9]{8}_[0-9]{4})_([0-9]{3}).*$"
match = re.match(file_regex,
                 os.path.basename(input_file).replace('-', '_'))
if not match:
    print(f"Could not extract time information from filename: {input_file} using regex {file_
↪regex}")
    sys.exit(1)

init = dt.datetime.strptime(match.group(1), '%Y%m%d_%H%M')
lead = int(match.group(2))
valid = init + dt.timedelta(hours=lead)

print("Data Shape: " + repr(met_data.shape))
print("Data Type:  " + repr(met_data.dtype))

print(valid)
```

(continues on next page)

(continued from previous page)

```
print(init)
print(lead)


attrs = {
   'valid': valid.strftime("%Y%m%d_%H%M%S"),
   'init':  init.strftime("%Y%m%d_%H%M%S"),
   'lead':  str(int(lead)),
   'accum': '00',

   'name':      'ivt',
   'long_name': 'integrated_vapor_transport',
   'level':     'Surface',
   'units':     'UNKNOWN',

   'grid': {
       'name': 'Global 0.5 Degree',
       'type' :   'LatLon',
       'lat_ll' : -90.0,
       'lon_ll' : 0.0,
       'delta_lat' :   0.5,
       'delta_lon' :   0.5,
       'Nlat' :      361,
       'Nlon' :      720,
   }
}
```

parm/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesB

```
# This script calculates potential vorticity (PV) from variables found in the GFS forecast␣
↪model grib files. This script is originally from Taylor Mandelbaum, SBU.
# Adjustments have been made by Lindsay Blank, NCAR.
# July 2020
##############################################################################################
↪######

import sys
import os
import re
import datetime as dt
from metpy import calc as mpcalc
from metpy.units import units
import xarray as xr
import cfgrib

##############################################################################################
```

(continues on next page)

(continued from previous page)

```
↪######

def pv(input_file):

    # Vars
    grib_vars = ['t','u','v']

    # Load a list of datasets, one for each variable we want
    ds_list = [cfgrib.open_datasets(input_file,backend_kwargs={'filter_by_keys':{'typeOfLevel
↪':'isobaricInhPa','shortName':v},'indexpath':''}) for v in grib_vars]

    # Flatten the list of lists to a single list of datasets
    ds_flat = [x.sel(isobaricInhPa=x.isobaricInhPa[x.isobaricInhPa>=100.0].values) for ds in␣
↪ds_list for x in ds]

    # Merge the variables into a single dataset
    ds = xr.merge(ds_flat)

    # Add pressure
    ds['p'] = xr.DataArray(ds.isobaricInhPa.values,dims=['isobaricInhPa'],coords={
↪'isobaricInhPa':ds.isobaricInhPa.values},attrs={'units':'hPa'}).broadcast_like(ds['t'])

    # Calculate potential temperature
    ds['theta'] = mpcalc.potential_temperature(ds['p'].metpy.convert_units('Pa'),ds['t'])

    # Compute baroclinic PV
    ds['pv'] = mpcalc.potential_vorticity_baroclinic(ds['theta'],ds['p'].metpy.convert_units(
↪'Pa'),ds['u'],ds['v'],latitude=ds.latitude)/(1.0e-6)

    met_data = ds['pv'].sel(isobaricInhPa=slice(float(os.environ.get('PV_LAYER_MAX_PRESSURE',
↪1000.0)),float(os.environ.get('PV_LAYER_MIN_PRESSURE',100.0)))).mean(axis=0).values

    return met_data

################################################################################################
↪######

input_file = os.path.expandvars(sys.argv[1])

data = pv(input_file) #Call function to calculate PV

met_data = data
met_data = met_data.astype('float64')

print("max", data.max())
```

(continues on next page)

(continued from previous page)

```
print("min", data.min())

# Automatically fill out time information from input file.
file_regex = r"^.*([0-9]{8}_[0-9]{4})_([0-9]{3}).*$"
match = re.match(file_regex, os.path.basename(input_file).replace('-', '_'))
if not match:
    print(f"Could not extract time information from filename: {input_file} using regex {file_
↪regex}")
    sys.exit(1)

init = dt.datetime.strptime(match.group(1), '%Y%m%d_%H%M')
lead = int(match.group(2))
valid = init + dt.timedelta(hours=lead)

print("Data Shape: " + repr(met_data.shape))
print("Data Type:  " + repr(met_data.dtype))

print(valid)
print(init)
print(lead)

attrs = {
        'valid': valid.strftime("%Y%m%d_%H%M%S"),
        'init':  init.strftime("%Y%m%d_%H%M%S"),
        'lead':  str(int(lead)),
        'accum': '00',

        'name':      'pv',
        'long_name': 'potential_vorticity',
        'level':     'Surface',
        'units':     'PV Units',

        'grid': {
            'name': 'Global 0.5 Degree',
            'type' :   'LatLon',
            'lat_ll' : -90.0,
            'lon_ll' : 0.0,
            'delta_lat' :   0.5,
            'delta_lon' :   0.5,
            'Nlat' :      361,
            'Nlon' :      720,
            }
        }
```

parm/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesB

```
# This script calculates potential vorticity (PV) from variables found in the GFS forecast␣
↪model grib files. This script is originally from Taylor Mandelbaum, SBU.
# Adjustments have been made by Lindsay Blank, NCAR.
# July 2020
##########################################################################################
↪######

import sys
import os
import re
import datetime as dt
from metpy import calc as mpcalc
from metpy.units import units
import xarray as xr
import cfgrib

##########################################################################################
↪######

def sept(input_file):

    # Vars
    grib_vars = ['t']

    # Load a list of datasets, one for each variable we want
    ds_list = [cfgrib.open_datasets(input_file,backend_kwargs={'filter_by_keys':{'typeOfLevel
↪':'isobaricInhPa','shortName':v},'indexpath':''}) for v in grib_vars]

    # Flatten the list of lists to a single list of datasets
    ds_flat = [x.sel(isobaricInhPa=x.isobaricInhPa[x.isobaricInhPa>=100.0].values) for ds in␣
↪ds_list for x in ds]

    # Merge the variables into a single dataset
    ds = xr.merge(ds_flat)

    # Add pressure
    ds['p'] = xr.DataArray(ds.isobaricInhPa.values,dims=['isobaricInhPa'],coords={
↪'isobaricInhPa':ds.isobaricInhPa.values},attrs={'units':'hPa'}).broadcast_like(ds['t'])

    # Calculate saturation equivalent potential temperature
    ds['sept'] = mpcalc.saturation_equivalent_potential_temperature(ds['p'].metpy.convert_
↪units('Pa'),ds['t'])

    met_data = ds['sept'].sel(isobaricInhPa=slice(float(os.environ.get('SEPT_LAYER_MAX_
↪PRESSURE',1000.0)),float(os.environ.get('SEPT_LAYER_MIN_PRESSURE',100.0)))).mean(axis=0).
↪values
```

(continues on next page)

(continued from previous page)

```
    return met_data

##########################################################################################
↪######

input_file = os.path.expandvars(sys.argv[1])

data = sept(input_file) #Call function to calculate PV

met_data = data
met_data = met_data.astype('float64')

print("max", data.max())
print("min", data.min())

# Automatically fill out time information from input file.
file_regex = r"^.*([0-9]{8}_[0-9]{4})_([0-9]{3}).*$"
match = re.match(file_regex, os.path.basename(input_file).replace('-', '_'))
if not match:
    print(f"Could not extract time information from filename: {input_file} using regex {file_
↪regex}")
    sys.exit(1)

init = dt.datetime.strptime(match.group(1), '%Y%m%d_%H%M')
lead = int(match.group(2))
valid = init + dt.timedelta(hours=lead)

print("Data Shape: " + repr(met_data.shape))
print("Data Type:  " + repr(met_data.dtype))

print(valid)
print(init)
print(lead)

attrs = {
        'valid': valid.strftime("%Y%m%d_%H%M%S"),
        'init':  init.strftime("%Y%m%d_%H%M%S"),
        'lead':  str(int(lead)),
        'accum': '00',

        'name':      'sept',
        'long_name': 'saturation_equivalent_potential_temperature',
        'level':     'Surface',
        'units':     'K',
```

(continues on next page)

(continued from previous page)

```
        'grid': {
            'name': 'Global 0.5 Degree',
            'type' :   'LatLon',
            'lat_ll' : -90.0,
            'lon_ll' : 0.0,
            'delta_lat' :   0.5,
            'delta_lon' :   0.5,
            'Nlat' :      361,
            'Nlon' :      720,
            }
        }
```

parm/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesB

```
# This script is a combination of two scripts originally from Taylor Mandelbaum, SBU.
# Adjustments have been made by Lindsay Blank, NCAR.
# May 2020
##########################################################################################
↪######

import pygrib
import numpy as np
import sys
import os
import re
import datetime as dt
import metpy.calc as mc

##########################################################################################
↪######

def ivt(input_file):
    grbs = pygrib.open(input_file)
    g = 9.81    # Setting gravity constant
    print(input_file)
    grbs.rewind()

    # Initialize variable arrays
    levs = [] # Levels
    q    = [] # Specific humidity
    hgt  = [] # Geopotential height
    temp = [] # Temperature
    u    = [] # u-wind
    v    = [] # v-wind
```

(continues on next page)

(continued from previous page)

```
    # First obtain the levels we will use
    # These are in hPa in the file, so directly compare with user supplied min/max
    levs = sorted(set([grb.level for grb in grbs if float(grb.level) >= float(os.environ.get(
↪'IVT_LAYER_MIN_PRESSURE',100.0)) and float(grb.level) <= float(os.environ.get('IVT_LAYER_
↪MAX_PRESSURE',1000.0))]))

    # Fill in variable arrays from input file.
    grbs.rewind()
    for grb in grbs:
        if not grb.level in levs:
            continue
        elif np.logical_and('v-' in grb.parameterName,grb.typeOfLevel=='isobaricInhPa'):
            v.append(grb.values)
        elif np.logical_and('u-' in grb.parameterName,grb.typeOfLevel=='isobaricInhPa'):
            u.append(grb.values)
        elif np.logical_and('Temperature' in grb.parameterName,grb.typeOfLevel==
↪'isobaricInhPa'):
            temp.append(grb.values)
        elif np.logical_and('Geopotential' in grb.parameterName,grb.typeOfLevel==
↪'isobaricInhPa'):
            hgt.append(grb.values)
        elif np.logical_and('Specific' in grb.parameterName,grb.typeOfLevel=='isobaricInhPa
↪'):
            q.append(grb.values)

    temp = np.array(temp)
    hgt  = np.array(hgt)
    u    = np.array(u)
    v    = np.array(v)

    grbs.rewind()

    # If we didn't find specific humidity, look for relative humidity.
    if len(q) == 0:
        for grb in grbs:
            if not grb.level in levs:
                continue
            if np.logical_and('Relative' in grb.parameterName,grb.typeOfLevel=='isobaricInhPa
↪'):
                q.append(grb.values)

        levs = np.array(levs)
        # Clausius-Clapeyron time
        es = 610.78*np.exp((17.67*(temp-273.15)/(temp-29.65))) # Calculate saturation vapor␣
↪pressure
```

(continues on next page)

(continued from previous page)

```
        e = es*(np.array(q)/100) # Calculate vapor pressure
        w = 0.622*es/(levs[:,None,None]*100) # Calculate water vapor
        q = w/(w+1) # Calculate specific humidity
    q = np.array(q)

    uv = np.sqrt(u**2+v**2) # Calculate wind
    mflux_total = np.sum(q,axis=0)*(1/g)*np.mean(uv,axis=0)*(np.max(levs)-np.min(levs))
↪#calculate mass flux
    met_data = mflux_total.copy() #Pass mass flux to be used by MET tools
    print(np.max(met_data))
    #np.save('{}.npy'.format(sys.argv[1]),mflux_total)
    grbs.close()

    return met_data

################################################################################################
↪######

input_file = os.path.expandvars(sys.argv[1])

data = ivt(input_file) #Call function to calculate IVT

met_data = data
met_data = met_data.astype('float64')

# Automatically fill out time information from input file.
for token in os.path.basename(input_file).replace('-', '_').split('_'):
   if(re.search("[0-9]{8,8}", token)):
       ymd = dt.datetime.strptime(token[0:8],"%Y%m%d")
   elif(re.search("^[0-9]{4}$", token)):
       hh  = int(token[0:2])
   elif(re.search("^[0-9]{3}$", token)):
       day = int(token.replace("", ""))

print("Data Shape: " + repr(met_data.shape))
print("Data Type:  " + repr(met_data.dtype))

# GFS Analysis
valid  = ymd  + dt.timedelta(hours=hh)
init = valid

attrs = {
   'valid': valid.strftime("%Y%m%d_%H%M%S"),
   'init':  init.strftime("%Y%m%d_%H%M%S"),
   'lead':  '00',
```

(continues on next page)

(continued from previous page)

```
   'accum': '00',

   'name':      'ivt',
   'long_name': 'integrated_vapor_transport',
   'level':     'Surface',
   'units':     'UNKNOWN',

   'grid': {
       'name': 'Global 0.5 Degree',
       'type' :   'LatLon',
       'lat_ll' : -90.0,
       'lon_ll' : 0.0,
       'delta_lat' :   0.5,
       'delta_lon' :   0.5,
       'Nlat' :      361,
       'Nlon' :      720,
   }
}
```

parm/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesB

```
# This script calculates potential vorticity (PV) from variables found in the GFS analysis␣
↪model grib2 files. This script is originally from Taylor Mandelbaum, SBU.
# Adjustments have been made by Lindsay Blank, NCAR.
# July 2020
################################################################################################
↪######

import sys
import os
import re
import datetime as dt
from metpy import calc as mpcalc
from metpy.units import units
import xarray as xr
import cfgrib

################################################################################################
↪######

def pv(input_file):

    # Vars
    grib_vars = ['t','u','v']
```

(continues on next page)

(continued from previous page)

```
    # Load a list of datasets, one for each variable we want
    ds_list = [cfgrib.open_datasets(input_file,backend_kwargs={'filter_by_keys':{'typeOfLevel
↪':'isobaricInhPa','shortName':v},'indexpath':''}) for v in grib_vars]

    # Flatten the list of lists to a single list of datasets
    ds_flat = [x.sel(isobaricInhPa=x.isobaricInhPa[x.isobaricInhPa>=100.0].values) for ds in␣
↪ds_list for x in ds]

    # Merge the variables into a single dataset
    ds = xr.merge(ds_flat)

    # Add pressure
    ds['p'] = xr.DataArray(ds.isobaricInhPa.values,dims=['isobaricInhPa'],coords={
↪'isobaricInhPa':ds.isobaricInhPa.values},attrs={'units':'hPa'}).broadcast_like(ds['t'])

    # Calculate potential temperature
    ds['theta'] = mpcalc.potential_temperature(ds['p'].metpy.convert_units('Pa'),ds['t'])

    # Compute baroclinic PV
    ds['pv'] = mpcalc.potential_vorticity_baroclinic(ds['theta'],ds['p'].metpy.convert_units(
↪'Pa'),ds['u'],ds['v'],latitude=ds.latitude)/(1.0e-6)

    met_data = ds['pv'].sel(isobaricInhPa=slice(float(os.environ.get('PV_LAYER_MAX_PRESSURE',
↪1000.0)),float(os.environ.get('PV_LAYER_MIN_PRESSURE',100.0)))).mean(axis=0).values

    return met_data

################################################################################################
↪######

input_file = os.path.expandvars(sys.argv[1])

data = pv(input_file) #Call function to calculate PV

met_data = data
met_data = met_data.astype('float64')

print("max", data.max())
print("min", data.min())

# Automatically fill out time information from input file.
for token in os.path.basename(input_file).replace('-', '_').split('_'):
    if(re.search("[0-9]{8,8}", token)):
        ymd = dt.datetime.strptime(token[0:8],"%Y%m%d")
    elif(re.search("^[0-9]{4}$", token)):
```

(continues on next page)

(continued from previous page)

```
        hh  = int(token[0:2])
    elif(re.search("^[0-9]{3}$", token)):
        day = int(token.replace("", ""))


print("Data Shape: " + repr(met_data.shape))
print("Data Type:  " + repr(met_data.dtype))

# GFS Analysis
valid  = ymd  + dt.timedelta(hours=hh)
init = valid
#lead, rem = divmod((valid-init).total_seconds(), 3600)


print(valid)
print(init)

attrs = {
    'valid': valid.strftime("%Y%m%d_%H%M%S"),
    'init':  init.strftime("%Y%m%d_%H%M%S"),
    'lead':  '00',
    'accum': '00',

    'name':      'pv',
    'long_name': 'potential_vorticity',
    'level':     'Surface',
    'units':     'PV Units',

    'grid': {
        'name': 'Global 0.5 Degree',
        'type' :   'LatLon',
        'lat_ll' : -90.0,
        'lon_ll' : 0.0,
        'delta_lat' :   0.5,
        'delta_lon' :   0.5,
        'Nlat' :      361,
        'Nlon' :      720,
    }
}
```

parm/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesB

```
# This script calculates potential vorticity (PV) from variables found in the GFS analysis␣
↪model grib2 files. This script is originally from Taylor Mandelbaum, SBU.
# Adjustments have been made by Lindsay Blank, NCAR.
# July 2020
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪######

import sys
import os
import re
import datetime as dt
from metpy import calc as mpcalc
from metpy.units import units
import xarray as xr
import cfgrib

##########################################################################################
↪######

def sept(input_file):

    # Vars
    grib_vars = ['t']

    # Load a list of datasets, one for each variable we want
    ds_list = [cfgrib.open_datasets(input_file,backend_kwargs={'filter_by_keys':{'typeOfLevel
↪':'isobaricInhPa','shortName':v},'indexpath':''}) for v in grib_vars]

    # Flatten the list of lists to a single list of datasets
    ds_flat = [x.sel(isobaricInhPa=x.isobaricInhPa[x.isobaricInhPa>=100.0].values) for ds in␣
↪ds_list for x in ds]

    # Merge the variables into a single dataset
    ds = xr.merge(ds_flat)

    # Add pressure
    ds['p'] = xr.DataArray(ds.isobaricInhPa.values,dims=['isobaricInhPa'],coords={
↪'isobaricInhPa':ds.isobaricInhPa.values},attrs={'units':'hPa'}).broadcast_like(ds['t'])

    # Calculate saturation equivalent potential temperature
    ds['sept'] = mpcalc.saturation_equivalent_potential_temperature(ds['p'].metpy.convert_
↪units('Pa'),ds['t'])

    met_data = ds['sept'].sel(isobaricInhPa=slice(float(os.environ.get('SEPT_LAYER_MAX_
↪PRESSURE',1000.0)),float(os.environ.get('SEPT_LAYER_MIN_PRESSURE',100.0)))).mean(axis=0).
↪values

    return met_data
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪######

input_file = os.path.expandvars(sys.argv[1])

data = sept(input_file) #Call function to calculate PV

met_data = data
met_data = met_data.astype('float64')

print("max", data.max())
print("min", data.min())

# Automatically fill out time information from input file.
for token in os.path.basename(input_file).replace('-', '_').split('_'):
    if(re.search("[0-9]{8,8}", token)):
        ymd = dt.datetime.strptime(token[0:8],"%Y%m%d")
    elif(re.search("^[0-9]{4}$", token)):
        hh  = int(token[0:2])
    elif(re.search("^[0-9]{3}$", token)):
        day = int(token.replace("", ""))


print("Data Shape: " + repr(met_data.shape))
print("Data Type:  " + repr(met_data.dtype))

# GFS Analysis
valid  = ymd  + dt.timedelta(hours=hh)
init = valid
#lead, rem = divmod((valid-init).total_seconds(), 3600)


print(valid)
print(init)

attrs = {
    'valid': valid.strftime("%Y%m%d_%H%M%S"),
    'init':  init.strftime("%Y%m%d_%H%M%S"),
    'lead':  '00',
    'accum': '00',

    'name':      'sept',
    'long_name': 'saturation_equivalent_potential_temperature',
    'level':     'Surface',
    'units':     'K',
```

(continues on next page)

(continued from previous page)

```
    'grid': {
        'name': 'Global 0.5 Degree',
        'type' :   'LatLon',
        'lat_ll' : -90.0,
        'lon_ll' : 0.0,
        'delta_lat' :   0.5,
        'delta_lon' :   0.5,
        'Nlat' :      361,
        'Nlon' :      720,
    }
}
```

Running METplus

This use case can be run two ways:

1) Passing in TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead_PyEmbed_Multiple_Diagnostics.co then a user-specific system configuration file:

```
run_metplus.py \
/path/to/METplus/parm/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_
↪fcstGFS_obsGFS_FeatureRelative_SeriesByLead_PyEmbed_Multiple_Diagnostics.conf \
/path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead_PyEmbed_Multiple_Diagnostics.conf:

```
run_metplus.py \
/path/to/METplus/parm/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_
↪fcstGFS_obsGFS_FeatureRelative_SeriesByLead_PyEmbed_Multiple_Diagnostics.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

  and for the [exe] section, you will need to define the location of NON-MET executables. If the executable is in the user's path, METplus will find it from the name. If the executable is not in the path, specify the full path to the executable here (i.e. CONVERT = /usr/bin/convert) The following executables are required for performing series analysis use cases:

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y

[exe]
CONVERT = /path/to/convert
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in subdirectories of the 'series_analysis_lead' directory (relative to **OUTPUT_BASE**):

- series_animate
- series_F090
- series_F096
- series_F102
- series_F108
- series_F114

The series_animate directory contains the animations of the series analysis in .gif format for all variable, level, and statistics combinations:

series_animate_<varname>_<level>_<stat>.gif

The series_FHHH directories contains files that have the following format:

ANLY_FILES_FHHH

FCST_ASCII_FILES_FHHH

series_FHHH_<varname>_<level>_<stat>.png

series_FHHH_<varname>_<level>_<stat>.ps

series_FHHH_<varname>_<level>_<stat>.nc

Where:

HHH is the forecast hour/lead time in hours

varname is the variable of interest, as specified in the METplus series_by_lead_all_fhrs config file

level is the level of interest, as specified in the METplus series_by_lead_all_fhrs config file

stat is the statistic of interest, as specified in the METplus series_by_lead_all_fhrs config file.

Keywords

Note:

- TCPairsToolUseCase
- SeriesByLeadUseCase
- TCStatToolUseCase
- RegridDataPlaneToolUseCase
- PyEmbedIngestToolUseCase
- MediumRangeAppUseCase
- SeriesAnalysisUseCase
- GRIB2FileUseCase
- FeatureRelativeUseCase
- SBUOrgUseCase
- DiagnosticsUseCase
- RuntimeFreqUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/medium_range-TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_Se

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.7.6 Point-Stat: Standard Verification of Global Upper Air

model_applications/medium_range/PointStat_fcstGFS_obsGDAS _UpperAir_MultiField _PrepBufr.conf

Scientific Objective

To provide useful statistical information on the relationship between observation data in point format to a gridded forecast. These values can be used to assess the skill of the prediction. Statistics are stored as partial sums to save space and Stat-Analysis must be used to compute the Continuous Statistics.

Datasets

Forecast: GFS temperature, u-wind component, v-wind component, and height
Observation: GDAS prepBURF data

Location: Click here for the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1331) section for more information.

METplus Components

This use case utilizes the METplus PB2NC wrapper to convert PrepBUFR point observations to NetCDF format and then compare them to gridded forecast data using PointStat.

METplus Workflow

PB2NC and PointStat are the tools called in this example. It processes the following run times:

Valid: 2017-06-01 0Z
Valid: 2017-06-02 0Z
Valid: 2017-06-03 0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/medium_range/PointStat_fcstGFS_obsGDAS_UpperAir_MultiField_PrepBufr.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/medium_range/
↪PointStat_fcstGFS_obsGDAS_UpperAir_MultiField_PrepBufr.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PB2NC, PointStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20170601
VALID_END = 20170603
VALID_INCREMENT = 86400

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

PB2NC_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_obs/prepbufr/gdas
PB2NC_INPUT_TEMPLATE = prepbufr.gdas.{valid?fmt=%Y%m%d%H}

PB2NC_OUTPUT_DIR = {OUTPUT_BASE}/gdas/upper_air
PB2NC_OUTPUT_TEMPLATE = prepbufr.gdas.{valid?fmt=%Y%m%d%H}.nc

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_obs/gfs
FCST_POINT_STAT_INPUT_TEMPLATE = pgbf{lead?fmt=%HH}.gfs.{init?fmt=%Y%m%d%H}

OBS_POINT_STAT_INPUT_DIR = {PB2NC_OUTPUT_DIR}
OBS_POINT_STAT_INPUT_TEMPLATE = {PB2NC_OUTPUT_TEMPLATE}

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/{OBTYPE}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = gfs
OBTYPE = gdas

BOTH_VAR1_NAME = TMP
BOTH_VAR1_LEVELS = P1000, P925, P850, P700, P500, P400, P300, P250, P200, P150, P100, P50,␣
↪P20, P10

BOTH_VAR2_NAME = RH
BOTH_VAR2_LEVELS = P1000, P925, P850, P700, P500, P400, P300

BOTH_VAR3_NAME = UGRD
BOTH_VAR3_LEVELS = P1000, P925, P850, P700, P500, P400, P300, P250, P200, P150, P100, P50,␣
↪P20, P10

BOTH_VAR4_NAME = VGRD
BOTH_VAR4_LEVELS = P1000, P925, P850, P700, P500, P400, P300, P250, P200, P150, P100, P50,␣
↪P20, P10

BOTH_VAR5_NAME = HGT
BOTH_VAR5_LEVELS = P1000, P950, P925, P850, P700, P500, P400, P300, P250, P200, P150, P100,␣
↪P50, P20, P10
```

(continues on next page)

(continued from previous page)

```
###
# PB2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pb2nc
###

# For both pb2nc and point_stat, the obs_window dictionary:
OBS_WINDOW_BEGIN = -2700
OBS_WINDOW_END = 2700

PB2NC_SKIP_IF_OUTPUT_EXISTS = True

PB2NC_QUALITY_MARK_THRESH = 3

PB2NC_PB_REPORT_TYPE = 120, 220, 221, 122, 222, 223, 224, 131, 133, 233, 153, 156, 157, 180,␣
↪280, 181, 182, 281, 282, 183, 284, 187, 287

PB2NC_LEVEL_CATEGORY = 0, 1, 4, 5, 6

PB2NC_GRID =
PB2NC_POLY =
PB2NC_STATION_ID =
PB2NC_MESSAGE_TYPE =

PB2NC_OBS_BUFR_VAR_LIST = QOB, TOB, ZOB, UOB, VOB, D_RH

PB2NC_TIME_SUMMARY_FLAG = False
PB2NC_TIME_SUMMARY_BEG = 000000
PB2NC_TIME_SUMMARY_END = 235959
PB2NC_TIME_SUMMARY_VAR_NAMES = PMO,TOB,TDO,UOB,VOB,PWO,TOCC
PB2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80


###
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###

POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST
POINT_STAT_INTERP_TYPE_METHOD = BILIN
POINT_STAT_INTERP_TYPE_WIDTH = 2

POINT_STAT_OUTPUT_FLAG_SL1L2 = STAT
POINT_STAT_OUTPUT_FLAG_VL1L2 = STAT
```

(continues on next page)

(continued from previous page)

```
POINT_STAT_REGRID_TO_GRID = G003
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2

POINT_STAT_GRID = FULL

POINT_STAT_POLY =
POINT_STAT_STATION_ID =

POINT_STAT_MESSAGE_TYPE = ADPUPA
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

PB2NCConfig_wrapped

Note: See the *PB2NC MET Configuration* (page 198) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// PB2NC configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// PrepBufr message type
//
${METPLUS_MESSAGE_TYPE}

//
// Mapping of message type group name to comma-separated list of values
// Derive PRMSL only for SURFACE message types
//
```

(continues on next page)

(continued from previous page)

```
message_type_group_map = [
   { key = "SURFACE"; val = "ADPSFC,SFCSHP,MSONET";               },
   { key = "ANYAIR";  val = "AIRCAR,AIRCFT";                      },
   { key = "ANYSFC";  val = "ADPSFC,SFCSHP,ADPUPA,PROFLR,MSONET"; },
   { key = "ONLYSF";  val = "ADPSFC,SFCSHP";                      }
];

//
// Mapping of input PrepBufr message types to output message types
//
message_type_map = [];

//
// PrepBufr station ID
//
${METPLUS_STATION_ID}


////////////////////////////////////////////////////////////////////////////////

//
// Observation time window
//
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observation retention regions
//
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observing location elevation
//
elevation_range = {
   beg =  -1000;
   end = 100000;
}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Observation types
//
//pb_report_type =
${METPLUS_PB_REPORT_TYPE}

in_report_type  = [];

instrument_type = [];

////////////////////////////////////////////////////////////////////////////////

//
// Vertical levels to retain
//
//level_range = {
${METPLUS_LEVEL_RANGE_DICT}

//level_category =
${METPLUS_LEVEL_CATEGORY}

////////////////////////////////////////////////////////////////////////////////

//
// BUFR variable names to retain or derive.
// If emtpy, process all available variables.
//
${METPLUS_OBS_BUFR_VAR}

////////////////////////////////////////////////////////////////////////////////

//
// Mapping of BUFR variable name to GRIB name. The default map is defined at
// obs_prepbufr_map. This replaces/expends the default map.
//
//obs_bufr_map =
${METPLUS_OBS_BUFR_MAP}

// This map is for PREPBUFR. It will be added into obs_bufr_map.
// Please do not override this map.
//obs_prepbufr_map =


////////////////////////////////////////////////////////////////////////////////

//quality_mark_thresh =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_QUALITY_MARK_THRESH}

event_stack_flag    = TOP;

////////////////////////////////////////////////////////////////////////////////
//
// Time periods for the summarization
//
${METPLUS_TIME_SUMMARY_DICT}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

//version = "V9.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

PointStatConfig_wrapped

Note: See the *PointStat MET Configuration* (page 218) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
```

(continues on next page)

(continued from previous page)

```
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
```

(continues on next page)

(continued from previous page)

```
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in PointStat_fcstGFS_obsGDAS_UpperAir_MultiField_PrepBufr.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪PointStat_fcstGFS_obsGDAS_UpperAir_MultiField_PrepBufr.conf -c /path/to/user_system.
↪conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PointStat_fcstGFS_obsGDAS_UpperAir_MultiField_PrepBufr.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪PointStat_fcstGFS_obsGDAS_UpperAir_MultiField_PrepBufr.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in gdas (relative to **OUTPUT_BASE**) and will contain the following files:

- point_stat_000000L_20170601_000000V.stat
- point_stat_000000L_20170602_000000V.stat
- point_stat_000000L_20170603_000000V.stat

Keywords

Note:

- PB2NCToolUseCase
- PointStatToolUseCase
- MediumRangeAppUseCase
- GRIBFileUseCase
- prepBUFRFileUseCase
- NOAAEMCOrgUseCase
- RegriddinginToolUseCase
- ObsTimeSummaryUseCase
- ClimatologyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/medium_range-PointStat_fcstGFS_obsGDAS_UpperAir_MultiField_PrepBufr

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.7.7 UserScript: Calculate the Difficulty Index

model_applications/medium_range/ UserScript_fcstGEFS _Difficulty_Index.conf

Scientific Objective

This use case calls the UserScript wrapper to run a user provided script that calculates the difficulty index for windspeed. This use case allows for the user to change a variety of variables needed to run the difficulty index (i.e. threshold start and units) so that user can run the script at different thresholds without needing to alter the code. This script run by the use case uses METcalcpy to provide the difficulty index calculation and METplotpy to provide the plotting capability.

The difficulty index was developed by the Naval Research Lab (NRL). The overall aim of the difficulty index is to graphically represent the expected difficulty of a decision based on a set of forecasts (ensemble) of, e.g., significant wave height as a function of space and time. There are two basic factors that can make a decision difficult. The first factor is the proximity of the ensemble mean forecast to a decision threshold, e.g. 12 ft seas. If the ensemble mean is either much lower or much higher than the threshold, the decision is easier; if it is closer to the threshold, the decision is harder. The second factor is the forecast precision, or ensemble spread. The greater the spread around the ensemble mean, the more likely it is that there will be ensemble members both above and below the decision threshold, making the decision harder. (A third factor that we will not address here is undiagnosed systematic error, which adds uncertainty in a similar way to ensemble spread.) The challenge is combining these factors into a continuous function that allows the user to assess relative risk.

Datasets

This use case calculates the difficulty index for windspeed using NCEP GEFS ensemble data. The data is composed of 30 ensemble members that have been compiled and compressed into one .npz file.

- Variables required to calculate the difficulty index: Levels required: 10-m #. v- component of wind #. u- component of wind #. Windspeed #. Latitude #. Longitude
- Forecast dataset: NCEP GEFS 30 member Ensemble - Initialization date: 20191208 - Initialization hours: 12 UTC - Lead times: 60 - Format: Grib2 - Resolution: 0.5 degree

METplus Components

This use case runs the UserScript wrapper tool to run a user provided script, in this case, wind_difficulty_index.py.

METplus Workflow

This use case loops by process which means that each tool is run for all times before moving to the next tool. The tool order is as follows:

UserScript

This example loops by initialization time (with begin, end, and increment as specified in the METplus UserScript_fcstGEFS_Difficulty_Index.conf file).

1 initialization time will be run over 1 lead time:

Init: 20201208_12Z
Forecast lead: 60

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/medium_range/UserScript_fcstGEFS_Difficulty_Index.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/medium_range/
↪UserScript_fcstGEFS_Difficulty_Index.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2020120812
INIT_END = 2020120812
INIT_INCREMENT = 12H

LEAD_SEQ =

USER_SCRIPT_CUSTOM_LOOP_LIST = nc

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_FOR_EACH


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

USER_SCRIPT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/diff_index
USER_SCRIPT_INPUT_TEMPLATE = {USER_SCRIPT_INPUT_DIR}/wndspd_GEFS_NorthPac_5dy_30mem_{init?
↪fmt=%Y%m%d%H}.npz

USER_SCRIPT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/medium_range/diff_index


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/model_applications/medium_range/UserScript_
↪fcstGEFS_Difficulty_Index/wind_difficulty_index.py


[user_env_vars]

# Difficulty index specific variables

DIFF_INDEX_INPUT_FILENAME = {USER_SCRIPT_INPUT_TEMPLATE}
```

(continues on next page)

(continued from previous page)

```
DIFF_INDEX_THRESH_START = 10.0

DIFF_INDEX_THRESH_END = 40.0

DIFF_INDEX_THRESH_STEP = 2.0

DIFF_INDEX_SAVE_THRESH_START = 20.0

DIFF_INDEX_SAVE_THRESH_STOP = 38.0

DIFF_INDEX_SAVE_THRESH_STEP = 2.0

DIFF_INDEX_UNITS = kn

DIFF_INDEX_FIG_FMT = png

DIFF_INDEX_FIG_BASENAME = {USER_SCRIPT_OUTPUT_DIR}/wndspd_GEFS_NorthPac_5dy_30mem_difficulty_
↪index
```

MET Configuration

There are no MET tools used in this use case.

Python Embedding

This use case uses a Python embedding script to read input data

parm/use_cases/model_applications/medium_range/UserScript_fcstGEFS_Difficulty_Index/wind_difficulty_index.py

```
#!/usr/bin/env python3

"""
Load fieldijn from npz file created with save_ensemble_data.py
helper function, compute ensemble mean and spread, compute
difficulty index for a set of thresholds, plot and save the results.
Author: Bill Campbell, NRL and Lindsay Blank, NCAR

Taken from original test_difficulty_index.py but replacing with METcalcpy and METplotpy.

"""
import os
import sys
import numpy as np
```

(continues on next page)

(continued from previous page)

```
import matplotlib.pyplot as plt
from metcalcpy.calc_difficulty_index import forecast_difficulty as di
from metcalcpy.calc_difficulty_index import EPS
from metcalcpy.piecewise_linear import PiecewiseLinear as plin
import metplotpy.plots.difficulty_index.mycolormaps as mcmap
from metplotpy.plots.difficulty_index.plot_difficulty_index import plot_field


def load_data(filename):
    """Load ensemble data from file"""
    loaded = np.load(filename)
    lats, lons = (loaded['lats'], loaded['lons'])
    fieldijn = np.ma.masked_invalid(
        np.ma.masked_array(
            data=loaded['data']))

    return lats, lons, fieldijn


def compute_stats(field):
    """Compute mean and std dev"""
    mu = np.mean(field, axis=-1)
    sigma = np.std(field, axis=-1, ddof=1)

    return mu, sigma

def compute_wind_envelope():
    """
    Computes piecewise linear envelope for winds in knots.

    Returns
    -------
    Piecewise linear object

    """
    # Envelope for version 6.1, the default
    xunits = 'kn'
    A6_1_name = "A6_1"
    A6_1_left = 0.0
    A6_1_right = 0.0
    A6_1_xlist = [5.0, 28.0, 34.0, 50.0]
    A6_1_ylist = [0.0, 1.5, 1.5, 0.0]
    Aplin =\
            plin(A6_1_xlist, A6_1_ylist, xunits=xunits,
                    right=A6_1_right, left=A6_1_left, name=A6_1_name)
```

(continues on next page)

(continued from previous page)

```
    return Aplin

def compute_difficulty_index(field, mu, sigma, thresholds, Aplin):
    """
    Compute difficulty index for an ensemble forecast given
    a set of thresholds, returning a dictionary of fields.
    """
    dij = {}
    for threshold in thresholds:
        dij[threshold] =\
            di(sigma, mu, threshold, field, Aplin=Aplin, sigma_over_mu_ref=EPS)

    return dij


def plot_difficulty_index(dij, lats, lons, thresholds, units):
    """
    Plot the difficulty index for a set of thresholds,
    returning a dictionary of figures
    """
    plt.close('all')
    myparams = {'figure.figsize': (8, 5),
                'figure.max_open_warning': 40}
    plt.rcParams.update(myparams)
    figs = {}
    cmap = mcmap.stoplight()
    for threshold in thresholds:
        if np.max(dij[threshold]) <= 1.0:
            vmax = 1.0
        else:
            vmax = 1.5
        figs[threshold] =\
            plot_field(dij[threshold],
                       lats, lons, vmin=0.0, vmax=vmax, cmap=cmap,
                       xlab='Longitude \u00b0E', ylab='Latitude',
                       clab='thresh={} {}'.format(threshold, units),
                       title='Forecast Decision Difficulty Index')

    return figs


def save_difficulty_figures(figs, save_thresh, units):
    """
    Save subset of difficulty index figures.
```

(continues on next page)

(continued from previous page)

```
    """
    fig_fmt = os.environ.get('DIFF_INDEX_FIG_FMT')
    fig_basename = os.environ.get('DIFF_INDEX_FIG_BASENAME')

    # create output directory if it does not already exist
    output_dir = os.path.dirname(fig_basename)
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    for thresh in save_thresh:
        thresh_str = '{:.2f}'.format(thresh).replace('.', '_')
        fig_name = (fig_basename + thresh_str +
                    '_' + units + '.' + fig_fmt)
        print('Saving {}...\n'.format(fig_name))
        figs[thresh].savefig(fig_name, format=fig_fmt)


def plot_statistics(mu, sigma, lats, lons, units='feet'):
    """Plot ensemble mean and spread, returning figure handles"""
    cmap = mcmap.spectral()
    mu_fig =\
        plot_field(mu, lats, lons, cmap=cmap, clab=units,
                   vmin=0.0, vmax=np.nanmax(mu),
                   xlab='Longitude \u00b0E',
                   ylab='Latitude',
                   title='Forecast Ensemble Mean')
    sigma_fig =\
        plot_field(sigma, lats, lons, cmap=cmap, clab=units,
                   vmin=0.0, vmax=np.nanmax(sigma),
                   xlab='Longitude \u00b0E',
                   ylab='Latitude',
                   title='Forecast Ensemble Std')

    return mu_fig, sigma_fig


def save_stats_figures(mu_fig, sigma_fig):
    """
    Save ensemble mean and spread figures.
    """

    fig_fmt = os.environ.get('DIFF_INDEX_FIG_FMT')
    fig_basename = os.environ.get('DIFF_INDEX_FIG_BASENAME')
    mu_name = fig_basename + 'mean.' + fig_fmt
    print('Saving {}...\n'.format(mu_name))
```

(continues on next page)

(continued from previous page)

```
    mu_fig.savefig(mu_name, format=fig_fmt)
    sigma_name = fig_basename + 'std.' + fig_fmt
    print('Saving {}...\n'.format(sigma_name))
    sigma_fig.savefig(sigma_name, format=fig_fmt)


def main():
    """
    Load fieldijn from npz file created with NCEP_test.py
    helper function, compute ensemble mean and spread, compute
    difficulty index for a set of thresholds, plot and save the results.
    """

    filename = os.environ.get('DIFF_INDEX_INPUT_FILENAME')
    lats, lons, fieldijn = load_data(filename)
    # Convert m/s to knots
    units = os.environ.get('DIFF_INDEX_UNITS')
    mps2kn = 1.94384
    fieldijn = mps2kn * fieldijn
    # Ensemble mean, std dev
    muij, sigmaij = compute_stats(fieldijn)
    # Windspeed envelope
    Aplin = compute_wind_envelope()
    # Difficulty index for a set of thresholds
    #thresholds = np.arange(os.environ.get('DIFF_INDEX_THRESH_START'), os.environ.get('DIFF_
↪INDEX_THRESH_END'), os.environ.get('DIFF_INDEX_THRESH_STEP'))
    start = float(os.environ.get('DIFF_INDEX_THRESH_START'))
    stop = float(os.environ.get('DIFF_INDEX_THRESH_END'))
    step = float(os.environ.get('DIFF_INDEX_THRESH_STEP'))
    thresholds = np.arange(start, stop, step)
    dij = compute_difficulty_index(fieldijn, muij, sigmaij, thresholds, Aplin=Aplin)
    # Plot and save difficulty index figures
    figs = plot_difficulty_index(dij, lats, lons, thresholds, units)
    save_start = float(os.environ.get('DIFF_INDEX_SAVE_THRESH_START'))
    save_stop = float(os.environ.get('DIFF_INDEX_SAVE_THRESH_STOP'))
    save_step = float(os.environ.get('DIFF_INDEX_SAVE_THRESH_STEP'))
    save_thresh = np.arange(save_start, save_stop, save_step)
    save_difficulty_figures(figs, save_thresh, units)
    # Plot and save ensemble mean, std_dev
    mu_fig, sigma_fig =\
        plot_statistics(muij, sigmaij, lats, lons, units=units)
    save_stats_figures(mu_fig, sigma_fig)


if __name__ == '__main__':
    main()
```

Running METplus

This use case can be run two ways:

1) Passing in UserScript_fcstGEFS_Difficulty_Index.conf, then a user-specific system configuration file:

```
run_metplus.py \
-c /path/to/METplus/parm/use_cases/model_applications/medium_range/UserScript_fcstGEFS_
↪Difficulty_Index.conf \
-c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_fcstGEFS_Difficulty_Index.conf:

```
run_metplus.py \
-c /path/to/METplus/parm/use_cases/model_applications/medium_range/UserScript_fcstGEFS_
↪Difficulty_Index.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

  and for the [exe] section, you will need to define the location of NON-MET executables. If the executable is in the user's path, METplus will find it from the name. If the executable is not in the path, specify the full path to the executable here (i.e. RM = /bin/rm) The following executables are required for performing series analysis use cases:

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y

[exe]
RM = /path/to/rm
CUT = /path/to/cut
TR = /path/to/tr
NCAP2 = /path/to/ncap2
CONVERT = /path/to/convert
NCDUMP = /path/to/ncdump
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in a directory relative to **OUTPUT_BASE**. There should be a list of files that have the following format:

wndspd_GEFS_NorthPac_5dy_30mem_difficulty_indexTHRESH_00_kn.png

Where THRESH isa number between DIFF_INDEX_SAVE_THRESH_START and DIFF_INDEX_SAVE_THRESH_STOP which are defined in UserScript_fcstGEFS_Difficulty_Index.conf.

Keywords

Note:

- UserScriptUseCase
- MediumRangeAppUseCase
- NRLOrgUseCase
- METcalcpyUseCase
- METplotpyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/medium_range-UserScript_fcstGEFS_Difficulty_Index.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.7.8 Multi_Tool (MTD): Feature Relative by Lead (with lead groupings)

model_applicaitons/medium_range/ MTD_SeriesAnalysis_fcstGFS _obsGFS_FeatureRelative _Series-ByLead.conf

Scientific Objective

Demonstrate the capability in the Feature Relative use case but using output from the MET MODE Time Domain (MTD) tool.

Datasets

Relevant information about the datasets that would be beneficial include:

- MODE Time Domain Forecast dataset: GFS
- Series-Analysis Forecast dataset: GFS
- MODE Time Domain Observation dataset: GFS Analysis
- Series-Analysis Observation dataset: GFS Analysis

METplus Components

This use case first runs MODE Time Domain and ExtractTiles wrappers to generate tiles of data centered on objects defined using MTD. The MET regrid_data_plane tool is used to regrid the data (GRIB1 or GRIB2 into netCDF). Next, a series analysis by lead time is performed on the results and plots (.ps and .png) are generated for all variable-level-stat combinations from the requested variables, levels, and requested statistics. The final results are aggregated into forecast hour groupings as specified by the start and end increment in the METplus configuration file, as well as labels to identify each forecast hour grouping.

METplus Workflow

The following tools are used for each run time:

MTD > RegridDataPlane (via ExtractTiles) > SeriesAnalysis

This example loops by forecast/lead time (with begin, end, and increment as specified in the METplus MTD_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.conf file). The following list of model initialization and forecast leads are processed in this use case:

Init: 20210712_00Z
Forecast lead: 6, 12, 18, 24, 30

Init: 20210712_06Z
Forecast lead: 6, 12, 18, 24, 30

Init: 20210712_12Z
Forecast lead: 6, 12, 18, 24, 30

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/medium_range/MTD_
↪SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = MTD, ExtractTiles, SeriesAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2021071200
INIT_END = 2021071212
INIT_INCREMENT = 6H

LEAD_SEQ = begin_end_incr(0,30,6)
```

(continues on next page)

(continued from previous page)

```
###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_MTD_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/MTD_SeriesAnalysis_fcstGFS_
↪obsGFS_FeatureRelative_SeriesByLead
FCST_MTD_INPUT_TEMPLATE= {init?fmt=%Y%m%d%H}/gfs.t{init?fmt=%H}z.pgrb2.1p00.f{lead?fmt=%HHH}

OBS_MTD_INPUT_DIR = {FCST_MTD_INPUT_DIR}
OBS_MTD_INPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}/gfs.t{valid?fmt=%H}z.pgrb2.1p00.f000

MTD_OUTPUT_DIR = {OUTPUT_BASE}/mtd
MTD_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}


EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS = no

EXTRACT_TILES_MTD_INPUT_DIR = {OUTPUT_BASE}/mtd
EXTRACT_TILES_MTD_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/mtd_{MODEL}_{FCST_VAR1_NAME}_vs_
↪{OBTYPE}_{OBS_VAR1_NAME}_{OBS_VAR1_LEVELS}_{init?fmt=%Y%m%d_%H%M%S}V_2d.txt

FCST_EXTRACT_TILES_INPUT_DIR = {FCST_MTD_INPUT_DIR}
FCST_EXTRACT_TILES_INPUT_TEMPLATE = {FCST_MTD_INPUT_TEMPLATE}

OBS_EXTRACT_TILES_INPUT_DIR = {FCST_MTD_INPUT_DIR}
OBS_EXTRACT_TILES_INPUT_TEMPLATE = {OBS_MTD_INPUT_TEMPLATE}

EXTRACT_TILES_OUTPUT_DIR = {OUTPUT_BASE}/extract_tiles
FCST_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/FCST_TILE_F{lead?fmt=%3H}_{MODEL}_
↪{init?fmt=%Y%m%d}_{init?fmt=%H}00_{lead?fmt=%3H}.nc
OBS_EXTRACT_TILES_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/OBS_TILE_F{lead?fmt=%3H}_{MODEL}_
↪{valid?fmt=%Y%m%d}_{valid?fmt=%H}00_000.nc


FCST_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = {FCST_EXTRACT_TILES_OUTPUT_TEMPLATE}

OBS_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = {OBS_EXTRACT_TILES_OUTPUT_TEMPLATE}

SERIES_ANALYSIS_TC_STAT_INPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}
```

(continues on next page)

(continued from previous page)

```
SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/series_analysis_lead
SERIES_ANALYSIS_OUTPUT_TEMPLATE = {label}/series_F{fcst_beg}_to_F{fcst_end}_{fcst_name}_
↪{fcst_level}.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = GFS
OBTYPE = GFS_ANLY

FCST_VAR1_NAME = PWAT
FCST_VAR1_LEVELS = L0

OBS_VAR1_NAME = PWAT
OBS_VAR1_LEVELS = L0


###
# MTD Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mtd
###

MTD_DESC = NA

MTD_SINGLE_RUN = False

FCST_MTD_CONV_RADIUS = 0
FCST_MTD_CONV_THRESH = gt60.0

OBS_MTD_CONV_RADIUS = 0
OBS_MTD_CONV_THRESH = gt60.0

MTD_REGRID_TO_GRID = NONE

MTD_MIN_VOLUME = 2000

MTD_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}_{CURRENT_FCST_
↪LEVEL}


###
```

(continues on next page)

(continued from previous page)

```
# ExtractTiles Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#extracttiles
###

EXTRACT_TILES_NLAT = 60
EXTRACT_TILES_NLON = 60

EXTRACT_TILES_DLAT = 0.5
EXTRACT_TILES_DLON = 0.5

EXTRACT_TILES_LON_ADJ = 15
EXTRACT_TILES_LAT_ADJ = 15


###
# SeriesAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#seriesanalysis
###

SERIES_ANALYSIS_BACKGROUND_MAP = no

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False

SERIES_ANALYSIS_REGRID_TO_GRID = OBS
SERIES_ANALYSIS_REGRID_METHOD = FORCE

SERIES_ANALYSIS_STAT_LIST = TOTAL, FBAR, OBAR, ME

SERIES_ANALYSIS_BLOCK_SIZE = 4000

SERIES_ANALYSIS_IS_PAIRED = True

SERIES_ANALYSIS_GENERATE_PLOTS = yes

SERIES_ANALYSIS_GENERATE_ANIMATIONS = yes

PLOT_DATA_PLANE_TITLE = {MODEL} series_F{fcst_beg}_to_F{fcst_end} Forecasts{nseries}, {stat}␣
↪for {fcst_name} {fcst_level}
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

MTDConfig_wrapped

Note: See the *MTD MET Configuration* (page 188) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE Time Domain configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

   //
   // Output model name to be written
   //

${METPLUS_MODEL}

   //
   // Output description to be written
   //

${METPLUS_DESC}

   //
   // Output observation type to be written
   //

${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

   //
   //  Verification grid
   //  May be set separately in each "field" entry
```

(continues on next page)

(continued from previous page)

```
   //
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

   //
   // Approximate grid resolution (km)
   //

grid_res = 4;

////////////////////////////////////////////////////////////////////////////////

   //
   // Forecast and observation fields to be verified
   //

fcst = {

   ${METPLUS_FCST_FILE_TYPE}

   ${METPLUS_FCST_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}

}

obs = {

   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
   //
   // Intensity percentile value to be written
   //

inten_perc_value = 99;

////////////////////////////////////////////////////////////////////////////////

   //
   //  Throw away 3D objects with volumes smaller than this
   //
${METPLUS_MIN_VOLUME}


////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine weights
   //

weight = {

   space_centroid_dist  = 1.0;

   time_centroid_delta  = 1.0;

   speed_delta          = 1.0;

   direction_diff       = 1.0;

   volume_ratio         = 1.0;

   axis_angle_diff      = 1.0;

   start_time_delta     = 1.0;

   end_time_delta       = 1.0;

}

////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine interest functions
```

(continues on next page)

(continued from previous page)

```
   //

interest_function = {

   space_centroid_dist = (

      (   0.0, 1.0 )
      (  50.0, 0.5 )
      ( 100.0, 0.0 )

   );

   time_centroid_delta = (

      ( -3.0, 0.0 )
      ( -2.0, 0.5 )
      ( -1.0, 0.8 )
      (  0.0, 1.0 )
      (  1.0, 0.8 )
      (  2.0, 0.5 )
      (  3.0, 0.0 )

   );

   speed_delta = (

      ( -10.0, 0.0 )
      (  -5.0, 0.5 )
      (   0.0, 1.0 )
      (   5.0, 0.5 )
      (  10.0, 0.0 )

   );

   direction_diff = (

      (   0.0, 1.0 )
      (  90.0, 0.0 )
      ( 180.0, 0.0 )

   );

   volume_ratio = (

      (  0.0, 0.0 )
```

(continues on next page)

(continued from previous page)

```
      (  0.5, 0.5 )
      (  1.0, 1.0 )
      (  1.5, 0.5 )
      (  2.0, 0.0 )

   );

   axis_angle_diff = (

      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 0.0 )

   );

   start_time_delta = (

      ( -5.0, 0.0 )
      ( -3.0, 0.5 )
      (  0.0, 1.0 )
      (  3.0, 0.5 )
      (  5.0, 0.0 )

   );

   end_time_delta = (

      ( -5.0, 0.0 )
      ( -3.0, 0.5 )
      (  0.0, 1.0 )
      (  3.0, 0.5 )
      (  5.0, 0.0 )

   );

}   //  interest functions


////////////////////////////////////////////////////////////////////////////////

   //
   // Total interest threshold for determining matches
   //

total_interest_thresh = 0.7;
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

   //
   // Output flags
   //

nc_output = {

   latlon       = true;
   raw          = true;
   object_id    = true;
   cluster_id   = true;

}

txt_output = {

   attributes_2d   = true;
   attributes_3d   = true;

}


////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

SeriesAnalysisConfig_wrapped

Note: See the *SeriesAnalysis MET Configuration* (page 232) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Series-Analysis configuration file.
```

(continues on next page)

(continued from previous page)

```
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
//
//desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
//obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

censor_thresh = [];
censor_val    = [];
//cat_thresh =
${METPLUS_CAT_THRESH}
cnt_thresh    = [ NA ];
cnt_logic     = UNION;

//
// Forecast and observation fields to be verified
//
fcst = {
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_CAT_THRESH}
   ${METPLUS_FCST_FIELD}
}
obs = {
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
```

(continues on next page)

(continued from previous page)

```
//mask = {
${METPLUS_MASK_DICT}

//
// Number of grid points to be processed concurrently.  Set smaller to use
// less memory but increase the number of passes through the data.
//
//block_size =
${METPLUS_BLOCK_SIZE}

//
// Ratio of valid matched pairs to compute statistics for a grid point
//
//vld_thresh =
${METPLUS_VLD_THRESH}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_stats = {
${METPLUS_OUTPUT_STATS_DICT}

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in MTD_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.conf, then a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/medium_range/MTD_
↪SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.conf
/path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in MTD_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.conf:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/medium_range/MTD_
↪SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

  If the 'convert' executable is not in the user's path, specify the full path to the executable here
- **CONVERT = /usr/bin/convert**

Example User Configuration File:

```
[config]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
CONVERT = /path/to/convert
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in series_analysis_lead directory relative to the **OUTPUT_BASE**, and in the following directories (relative to **OUTPUT_BASE**):

- series_FHHH
- series_animate

The *series_FHHH* subdirectory will contain files that have the following format:

OBS_FILES_FHHH

FCST_FILES_FHHH

series_Fhhh_to_FHHH_<varname>_<level>_<stat>.png

series_Fhhh_to_FHHH_<varname>_<level>_<stat>.ps

series_Fhhh_to_FHHH_<varname>_<level>_<stat>.nc

Where:

hhh is the starting forecast hour/lead time in hours

HHH is the ending forecast hour/lead time in hours

varname is the variable of interest, as specified in the METplus series_by_lead_all_fhrs config file

level is the level of interest, as specified in the METplus series_by_lead_all_fhrs config file

stat is the statistic of interest, as specified in the METplus series_by_lead_all_fhrs config file.

The series_animate directory contains the animations of the series analysis in .gif format for all variable, level, and statistics combinations:

series_animate_<varname>_<level>_<stat>.gif

Keywords

Note:

- MediumRangeAppUseCase
- TCPairsToolUseCase
- SeriesByLeadUseCase
- MTDToolUseCase
- RegridDataPlaneToolUseCase
- SeriesAnalysisUseCase

- GRIB2FileUseCase
- FeatureRelativeUseCase
- SBUOrgUseCase
- DiagnosticsUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/medium_range-MTD_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_Serie

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.7.9 Point-Stat: Standard Verification for CONUS Surface

model_applications/medium_range/PointStat_fcstGFS_obsNAM_Sfc _MultiField_PrepBufr.conf

Scientific Objective

To provide useful statistical information on the relationship between observation data in point format to a gridded forecast. These values can be used to assess the skill of the prediction. Statistics are store as partial sums to save space and Stat-Analysis must be used to compute Continuous statistics.

Datasets

Forecast: GFS temperature, u-wind component, v-wind component, and height
Observation: NAM prepBURF data

Location: Click here for the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1371) section for more information.

Data Source: Unknown

METplus Components

This use case utilizes the METplus PB2NC wrapper to convert PrepBUFR point observations to NetCDF format and then compare them to gridded forecast data using PointStat.

METplus Workflow

PB2NC and PointStat are the tools called in this example. It processes the following run times:

Valid: 2017-06-01 0Z

Valid: 2017-06-02 0Z

Valid: 2017-06-03 0Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/medium_range/PointStat_fcstGFS_obsNAM_Sfc_MultiField_PrepBufr.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/medium_range/
↪PointStat_fcstGFS_obsNAM_Sfc_MultiField_PrepBufr.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PB2NC, PointStat

###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
```

(continues on next page)

(continued from previous page)

```
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20170601
VALID_END = 20170603
VALID_INCREMENT = 86400

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

PB2NC_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_obs/prepbufr/nam
PB2NC_INPUT_TEMPLATE = nam.{da_init?fmt=%Y%m%d}/nam.t{da_init?fmt=%H}z.prepbufr.tm{offset?
↪fmt=%2H}

PB2NC_OUTPUT_DIR = {OUTPUT_BASE}/nam/conus_sfc
PB2NC_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/nam.{valid?fmt=%Y%m%d%H}.nc
PB2NC_SKIP_IF_OUTPUT_EXISTS = True

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_obs/gfs
FCST_POINT_STAT_INPUT_TEMPLATE = pgbf{lead?fmt=%HH}.gfs.{init?fmt=%Y%m%d%H}

OBS_POINT_STAT_INPUT_DIR = {PB2NC_OUTPUT_DIR}
OBS_POINT_STAT_INPUT_TEMPLATE = {PB2NC_OUTPUT_TEMPLATE}

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/{OBTYPE}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###
```

(continues on next page)

(continued from previous page)

```
MODEL = gfs
OBTYPE = nam

BOTH_VAR1_NAME = TMP
BOTH_VAR1_LEVELS = Z2

BOTH_VAR2_NAME = RH
BOTH_VAR2_LEVELS = Z2

BOTH_VAR3_NAME = DPT
BOTH_VAR3_LEVELS = Z2

BOTH_VAR4_NAME = UGRD
BOTH_VAR4_LEVELS = Z10

BOTH_VAR5_NAME = VGRD
BOTH_VAR5_LEVELS = Z10

BOTH_VAR6_NAME = TCDC
BOTH_VAR6_LEVELS = L0
FCST_VAR6_OPTIONS = GRIB_lvl_typ = 200;

BOTH_VAR7_NAME = PRMSL
BOTH_VAR7_LEVELS = Z0


###
# PB2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pb2nc
###

# For both pb2nc and point_stat, the obs_window dictionary:
OBS_WINDOW_BEGIN = -2700
OBS_WINDOW_END = 2700

PB2NC_QUALITY_MARK_THRESH = 3

PB2NC_PB_REPORT_TYPE = 120, 220, 221, 122, 222, 223, 224, 131, 133, 233, 153, 156, 157, 180,␣
↪280, 181, 182, 281, 282, 183, 284, 187, 287

PB2NC_LEVEL_CATEGORY = 0, 1, 4, 5, 6

PB2NC_GRID =
PB2NC_POLY =
```

(continues on next page)

(continued from previous page)

```
PB2NC_STATION_ID =
PB2NC_MESSAGE_TYPE =

PB2NC_OBS_BUFR_VAR_LIST = PMO, TOB, TDO, UOB, VOB, PWO, TOCC, D_RH

PB2NC_TIME_SUMMARY_FLAG = False
PB2NC_TIME_SUMMARY_BEG = 000000
PB2NC_TIME_SUMMARY_END = 235959
PB2NC_TIME_SUMMARY_VAR_NAMES = PMO,TOB,TDO,UOB,VOB,PWO,TOCC
PB2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80


###
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###

POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST
POINT_STAT_INTERP_TYPE_METHOD = BILIN
POINT_STAT_INTERP_TYPE_WIDTH = 2

POINT_STAT_OUTPUT_FLAG_SL1L2 = STAT
POINT_STAT_OUTPUT_FLAG_VL1L2 = STAT

POINT_STAT_REGRID_TO_GRID = G104
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2

POINT_STAT_GRID = FULL
POINT_STAT_POLY =
POINT_STAT_STATION_ID =

POINT_STAT_MESSAGE_TYPE = ONLYSF
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

PB2NCConfig_wrapped

Note: See the *PB2NC MET Configuration* (page 198) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// PB2NC configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// PrepBufr message type
//
${METPLUS_MESSAGE_TYPE}

//
// Mapping of message type group name to comma-separated list of values
// Derive PRMSL only for SURFACE message types
//
message_type_group_map = [
   { key = "SURFACE"; val = "ADPSFC,SFCSHP,MSONET";               },
   { key = "ANYAIR";  val = "AIRCAR,AIRCFT";                      },
   { key = "ANYSFC";  val = "ADPSFC,SFCSHP,ADPUPA,PROFLR,MSONET"; },
   { key = "ONLYSF";  val = "ADPSFC,SFCSHP";                      }
];

//
// Mapping of input PrepBufr message types to output message types
//
message_type_map = [];

//
// PrepBufr station ID
//
${METPLUS_STATION_ID}


////////////////////////////////////////////////////////////////////////////////

//
// Observation time window
//
${METPLUS_OBS_WINDOW_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Observation retention regions
//
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observing location elevation
//
elevation_range = {
   beg =  -1000;
   end = 100000;
}

////////////////////////////////////////////////////////////////////////////////

//
// Observation types
//
//pb_report_type =
${METPLUS_PB_REPORT_TYPE}

in_report_type  = [];

instrument_type = [];

////////////////////////////////////////////////////////////////////////////////

//
// Vertical levels to retain
//
//level_range = {
${METPLUS_LEVEL_RANGE_DICT}

//level_category =
${METPLUS_LEVEL_CATEGORY}

////////////////////////////////////////////////////////////////////////////////

//
// BUFR variable names to retain or derive.
// If emtpy, process all available variables.
```

(continues on next page)

(continued from previous page)

```
//
${METPLUS_OBS_BUFR_VAR}

////////////////////////////////////////////////////////////////////////////////

//
// Mapping of BUFR variable name to GRIB name. The default map is defined at
// obs_prepbufr_map. This replaces/expends the default map.
//
//obs_bufr_map =
${METPLUS_OBS_BUFR_MAP}

// This map is for PREPBUFR. It will be added into obs_bufr_map.
// Please do not override this map.
//obs_prepbufr_map =


////////////////////////////////////////////////////////////////////////////////

//quality_mark_thresh =
${METPLUS_QUALITY_MARK_THRESH}

event_stack_flag    = TOP;

////////////////////////////////////////////////////////////////////////////////
//
// Time periods for the summarization
//
${METPLUS_TIME_SUMMARY_DICT}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

//version = "V9.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

PointStatConfig_wrapped

Note: See the *PointStat MET Configuration* (page 218) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;
```

(continues on next page)

(continued from previous page)

```
//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}
```

(continues on next page)

(continued from previous page)

```
//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in PointStat_fcstGFS_obsNAM_Sfc_MultiField_PrepBufr.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪PointStat_fcstGFS_obsNAM_Sfc_MultiField_PrepBufr.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PointStat_fcstGFS_obsNAM_Sfc_MultiField_PrepBufr.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪PointStat_fcstGFS_obsNAM_Sfc_MultiField_PrepBufr.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in nam (relative to **OUTPUT_BASE**) and will contain the following files:

- point_stat_000000L_20170601_000000V.stat
- point_stat_000000L_20170602_000000V.stat
- point_stat_000000L_20170603_000000V.stat

Keywords

Note:

- PB2NCToolUseCase
- MediumRangeAppUseCase
- PointStatToolUseCase
- GRIBFileUseCase
- prepBUFRFileUseCase
- NOAAEMCOrgUseCase
- RegriddinginToolUseCase
- ObsTimeSummaryUseCase
- ClimatologyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/medium_range-PointStat_fcstGFS_obsNAM_Sfc_MultiField_PrepBufr.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.7.10 Grid-Stat: Compute Anomaly Correlation using Climatology

model_applications/medium_range/GridStat_fcstGFS_obsGFS _climoNCEP_MultiField.conf

Scientific Objective

To provide useful statistical information on the relationship between observation data in gridded format to a gridded forecast. These values can be used to help correct model deviations from observed values.

Datasets

Forecast: GFS
Observation: GFS
climotology: NCEP
Location: Click here for the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1384) section for more information.
Data Source: Unknown

METplus Components

This use case utilizes the METplus GridStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool grid_stat if all required files are found. Then StatAnalysis is run on the GridStat output.

METplus Workflow

GridStat and StatAnalysis are the tools called in this example. It processes the following run times:

Valid: 2017-06-13 0Z
Forecast lead: 24 hour

Valid: 2017-06-13 0Z
Forecast lead: 48 hour

Valid: 2017-06-13 6Z
Forecast lead: 24 hour

Valid: 2017-06-13 6Z
Forecast lead: 48 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/medium_range/GridStat_fcstGFS_obsGFS_climoNCEP_MultiField.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/medium_range/
↪GridStat_fcstGFS_obsGFS_climoNCEP_MultiField.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###
```

(continues on next page)

(continued from previous page)

```
PROCESS_LIST = GridStat, StatAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2017061300
VALID_END = 2017061306
VALID_INCREMENT = 21600

LEAD_SEQ = 24, 48


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

GRID_STAT_CLIMO_MEAN_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_grid/
↪nwprod/fix
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE = cmean_1d.1959{valid?fmt=%m%d}

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_grid/gfs/fcst
FCST_GRID_STAT_INPUT_TEMPLATE = pgbf{lead?fmt=%.3H}.gfs.{init?fmt=%Y%m%d%H}

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_grid/gfs/obs
OBS_GRID_STAT_INPUT_TEMPLATE = pgbanl.gfs.{valid?fmt=%Y%m%d%H}

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_out/{MODEL}/anom
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H%M}/grid_stat

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {OUTPUT_BASE}/met_out/{MODEL1}/anom/*/grid_stat
```

(continues on next page)

(continued from previous page)

```
STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/gather_by_date/stat_analysis/grid2grid/anom
MODEL1_STAT_ANALYSIS_DUMP_ROW_TEMPLATE = {fcst_valid_hour?fmt=%H}Z/{MODEL1}/{MODEL1}_{valid?
↪fmt=%Y%m%d}.stat


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

BOTH_VAR1_NAME = TMP
BOTH_VAR1_LEVELS = P850, P500, P250

BOTH_VAR2_NAME = UGRD
BOTH_VAR2_LEVELS = P850, P500, P250

BOTH_VAR3_NAME = VGRD
BOTH_VAR3_LEVELS = P850, P500, P250

BOTH_VAR4_NAME = PRMSL
BOTH_VAR4_LEVELS = Z0


###
# GridStat Settings (optional)
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

MODEL = GFS
OBTYPE = ANLYS

GRID_STAT_GRID_WEIGHT_FLAG = COS_LAT

GRID_STAT_OUTPUT_PREFIX = {MODEL}_vs_{OBTYPE}

GRID_STAT_REGRID_TO_GRID = G002
GRID_STAT_REGRID_METHOD = BILIN
GRID_STAT_REGRID_WIDTH = 2

GRID_STAT_MASK_GRID = FULL
GRID_STAT_MASK_POLY = {INPUT_BASE}/model_applications/medium_range/poly/NHX.nc, {INPUT_BASE}/
↪model_applications/medium_range/poly/SHX.nc, {INPUT_BASE}/model_applications/medium_range/
↪poly/TRO.nc, {INPUT_BASE}/model_applications/medium_range/poly/PNA.nc
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_CLIMO_CDF_WRITE_BINS = False

GRID_STAT_OUTPUT_FLAG_SAL1L2 = STAT
GRID_STAT_OUTPUT_FLAG_VAL1L2 = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE

GRID_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST

GRID_STAT_CLIMO_MEAN_REGRID_METHOD = BILIN
GRID_STAT_CLIMO_MEAN_REGRID_WIDTH = 2
GRID_STAT_CLIMO_MEAN_DAY_INTERVAL = 1

GRID_STAT_MET_CONFIG_OVERRIDES = climo_mean = fcst;


###
# StatAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#statanalysis
###

STAT_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/STATAnalysisConfig_wrapped

MODEL1 = GFS
MODEL1_OBTYPE = ANLYS

STAT_ANALYSIS_JOB_NAME = filter
STAT_ANALYSIS_JOB_ARGS = -dump_row [dump_row_file]

MODEL_LIST = {MODEL1}
DESC_LIST =
FCST_LEAD_LIST =
OBS_LEAD_LIST =
FCST_VALID_HOUR_LIST = 00, 06
FCST_INIT_HOUR_LIST = 00, 06
OBS_VALID_HOUR_LIST =
OBS_INIT_HOUR_LIST =
FCST_VAR_LIST =
OBS_VAR_LIST =
FCST_UNITS_LIST =
OBS_UNITS_LIST =
```

(continues on next page)

(continued from previous page)

```
FCST_LEVEL_LIST =
OBS_LEVEL_LIST =
VX_MASK_LIST =
INTERP_MTHD_LIST =
INTERP_PNTS_LIST =
FCST_THRESH_LIST =
OBS_THRESH_LIST =
COV_THRESH_LIST =
ALPHA_LIST =
LINE_TYPE_LIST =

GROUP_LIST_ITEMS = FCST_INIT_HOUR_LIST
LOOP_LIST_ITEMS = FCST_VALID_HOUR_LIST, MODEL_LIST
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

GridStatConfig_wrapped

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}
```

(continues on next page)

(continued from previous page)

```
//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
```

(continues on next page)

(continued from previous page)

```
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
```

(continues on next page)

(continued from previous page)

```
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
```

(continues on next page)

(continued from previous page)

```
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

StatAnalysisConfig_wrapped

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// STAT-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Filtering input STAT lines by the contents of each column
//
${METPLUS_MODEL}
${METPLUS_DESC}

${METPLUS_FCST_LEAD}
${METPLUS_OBS_LEAD}

${METPLUS_FCST_VALID_BEG}
${METPLUS_FCST_VALID_END}
${METPLUS_FCST_VALID_HOUR}

${METPLUS_OBS_VALID_BEG}
${METPLUS_OBS_VALID_END}
${METPLUS_OBS_VALID_HOUR}

${METPLUS_FCST_INIT_BEG}
${METPLUS_FCST_INIT_END}
${METPLUS_FCST_INIT_HOUR}

${METPLUS_OBS_INIT_BEG}
${METPLUS_OBS_INIT_END}
${METPLUS_OBS_INIT_HOUR}

${METPLUS_FCST_VAR}
${METPLUS_OBS_VAR}

${METPLUS_FCST_UNITS}
${METPLUS_OBS_UNITS}

${METPLUS_FCST_LEVEL}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OBS_LEVEL}

${METPLUS_OBTYPE}

${METPLUS_VX_MASK}

${METPLUS_INTERP_MTHD}

${METPLUS_INTERP_PNTS}

${METPLUS_FCST_THRESH}
${METPLUS_OBS_THRESH}
${METPLUS_COV_THRESH}

${METPLUS_ALPHA}

${METPLUS_LINE_TYPE}

column = [];

weight = [];

////////////////////////////////////////////////////////////////////////////////

//
// Array of STAT-Analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
out_alpha = 0.05;

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// WMO mean computation logic
//
wmo_sqrt_stats   = [ "CNT:FSTDEV",  "CNT:OSTDEV",  "CNT:ESTDEV",
                     "CNT:RMSE",    "CNT:RMSFA",   "CNT:RMSOA",
                     "VCNT:FS_RMS", "VCNT:OS_RMS", "VCNT:RMSVE",
                     "VCNT:FSTDEV", "VCNT:OSTDEV" ];

wmo_fisher_stats = [ "CNT:PR_CORR", "CNT:SP_CORR",
                     "CNT:KT_CORR", "CNT:ANOM_CORR" ];

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;
vif_flag       = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstGFS_obsGFS_climoNCEP_MultiField.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪GridStat_fcstGFS_obsGFS_climoNCEP_MultiField.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstGFS_obsGFS_climoNCEP_MultiField.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/medium_range/
↪GridStat_fcstGFS_obsGFS_climoNCEP_MultiField.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in

parm/use_cases

- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in gather_by_date/stat_analysis/grid2grid/anom (relative to **OUTPUT_BASE**) and will contain the following files:

- 00Z/GFS/GFS_20170613.stat
- 06Z/GFS/GFS_20170613.stat

Keywords

Note:

- GridStatToolUseCase
- MediumRangeAppUseCase
- StatAnalysisToolUseCase
- GRIBFileUseCase
- NOAAEMCOrgUseCase
- RegriddinginTool
- ClimatologyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/medium_range-GridStat_fcstGFS_obsGFS_climoNCEP_MultiField.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.8 Planetary Boundary Layer

Planetary Boundary Layer (PBL) applications

7.2.16.8.1 GenVxMask and Point-Stat: Computing PBLH from AMDAR data using "Theta-increase" method

model_applications/pbl/PointStat_fcstHRRR_obsAMDAR_PBLH_PyEmbed.conf

Scientific Objective

The Planetary Boundary Layer Height (PBLH) arises from a complex interaction of lower atmosphere and surface processes, and is therefore a useful metric to evaluate models. This PointStat use case computes PBLH from AMDAR aircraft data using the "Theta-increase" method (Nielsen-Gammon et al., 2008, J. App. Met. Clim.), which computes PBLH by finding the lowest altitude in an aircraft profile that exceeds a specified increase in potential temperature from a base value. Generally this theta-increase (pt_delta) ranges from 1.0-2.5 K. The pt_delta, list of airports to process, and sounding are specified in the configuration script.

Datasets

Forecast: HRRR, RRFS (reads the "HPBL" grib2 field) Observation: AMDAR hourly 1-d netcdf files

METplus Components

This use case utilizes GenVxMask and the METplus PointStat tool to compare PBLH from AMDAR data to model output. The python embedding script "calc_amdar_pblh.py" computes PBLH and sends data MET via python embedding. The configuration file also filters output through static geographic masks generated by GenVxMask.

METplus Workflow

GenVxMask and PointStat are called in this example. The following run times are processed:

Valid: 2022-07-01_20Z
Forecast lead: 12 hour

GenVxMask input file is a two-row text file (met_mask_AIRPORT.txt): row 1: AIRPORT row 2: lat lon GenVxMask output file is a netcdf file w/ geographic radius part of the file name

(met_mask_AIRPORT_100km.nc) Input file provided in this example: (met_mask_DENVER.txt): row 1: DENVER row2: 39.856 -104.6764

```
# PointStat is run with Python embedding (calc_amdar_pblh.py).
```

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/pbl/PointStat_fcstHRRR_obsAMDAR_PBLH_PyEmbed.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/pbl/PointStat_
↪fcstHRRR_obsAMDAR_PBLH_PyEmbed.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GenVxMask, PointStat

###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2022070108
INIT_END = 2022070108
INIT_INCREMENT = 1H

LEAD_SEQ = 12
```

(continues on next page)

(continued from previous page)

```
# USER SETTINGS HERE
# CUSTOM_LOOP_LIST = DENVER, DALLAS, BOSTON, MINNEAPOLIS   [list of airports sepearated by␣
↪commas; each needs a mask file]
# PY_SOUNDING_FLAG = ALL, ASC, DESC  [only one value here]
# PY_PT_DELTA =  Potential Temperature delta setting (K) [usually 1.0-2.5]
# Valid_Time(YYYYMMDD_HHMMSS)
GEN_VX_MASK_AIRPORT_RADIUS_KM = 100
CUSTOM_LOOP_LIST = DENVER
PY_SOUNDING_FLAG = ALL
PY_PT_DELTA = 1.25
PY_VAL_TIME = {valid?fmt=%Y%m%d_%H0000}

###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

GEN_VX_MASK_INPUT_DIR = {INPUT_BASE}/model_applications/pbl/PointStat_fcstHRRR_obsAMDAR_PBLH_
↪PyEmbed
GEN_VX_MASK_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/pblh_22{init?fmt=%j}{init?fmt=%H}0000{lead?
↪fmt=%HH}
GEN_VX_MASK_INPUT_MASK_DIR = {INPUT_BASE}/model_applications/pbl/PointStat_fcstHRRR_obsAMDAR_
↪PBLH_PyEmbed
GEN_VX_MASK_INPUT_MASK_TEMPLATE = met_mask_{custom?fmt=%s}.txt

GEN_VX_MASK_OUTPUT_DIR = {INPUT_BASE}/model_applications/pbl/PointStat_fcstHRRR_obsAMDAR_
↪PBLH_PyEmbed
GEN_VX_MASK_OUTPUT_TEMPLATE = {OUTPUT_BASE}/gen_vx_mask_pblh/met_mask_{custom?fmt=%s}_{GEN_
↪VX_MASK_AIRPORT_RADIUS_KM}km.nc

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/pbl/PointStat_fcstHRRR_obsAMDAR_
↪PBLH_PyEmbed
FCST_POINT_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/pblh_22{init?fmt=%j}{init?fmt=%H}0000
↪{lead?fmt=%HH}

OBS_POINT_STAT_INPUT_DIR =
OBS_POINT_STAT_INPUT_TEMPLATE = PYTHON_NUMPY= {PARM_BASE}/use_cases/model_applications/pbl/
↪PointStat_fcstHRRR_obsAMDAR_PBLH_PyEmbed/calc_amdar_pblh.py {INPUT_BASE}/model_
↪applications/pbl/PointStat_fcstHRRR_obsAMDAR_PBLH_PyEmbed/22{valid?fmt=%j}{valid?fmt=%H}
↪00q.cdf

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/point_stat_pblh
POINT_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}
```

(continues on next page)

(continued from previous page)

```
POINT_STAT_CLIMO_MEAN_INPUT_DIR =
POINT_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

POINT_STAT_CLIMO_STDEV_INPUT_DIR =
POINT_STAT_CLIMO_STDEV_INPUT_TEMPLATE =


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

POINT_STAT_ONCE_PER_FIELD = False

FCST_VAR1_NAME = HPBL
FCST_VAR1_LEVELS = L0
FCST_VAR1_THRESH = <=0, >10000
OBS_VAR1_NAME = HPBL
OBS_VAR1_LEVELS = L0
OBS_VAR1_THRESH = <=0, >10000
OBS_VAR1_OPTIONS = set_attr_units = "m"


###
# GenVxMask Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#genvxmask
###

GEN_VX_MASK_SKIP_IF_OUTPUT_EXISTS = True
GEN_VX_MASK_OPTIONS = -type "circle" -thresh le{GEN_VX_MASK_AIRPORT_RADIUS_KM}

# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###

LOG_POINT_STAT_VERBOSITY = 4

POINT_STAT_CONFIG_FILE ={PARM_BASE}/met_config/PointStatConfig_wrapped

POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST
POINT_STAT_INTERP_TYPE_METHOD = BILIN
POINT_STAT_INTERP_TYPE_WIDTH = 2

POINT_STAT_OUTPUT_FLAG_SL1L2 = STAT
```

(continues on next page)

(continued from previous page)

```
POINT_STAT_OUTPUT_FLAG_VL1L2 = STAT
POINT_STAT_OUTPUT_FLAG_MPR = STAT

OBS_POINT_STAT_WINDOW_BEGIN = -3600
OBS_POINT_STAT_WINDOW_END = 3600

POINT_STAT_OFFSETS = 0

MODEL = HRRR

POINT_STAT_DESC = {PY_SOUNDING_FLAG}_{PY_PT_DELTA}
OBTYPE =

POINT_STAT_REGRID_TO_GRID = NONE
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2

POINT_STAT_OUTPUT_PREFIX = {custom?fmt=%s}_{PY_SOUNDING_FLAG}_{PY_PT_DELTA}

POINT_STAT_MASK_GRID =
POINT_STAT_MASK_POLY = {OUTPUT_BASE}/gen_vx_mask_pblh/met_mask_{custom?fmt=%s}_{GEN_VX_MASK_
↪AIRPORT_RADIUS_KM}km.nc
POINT_STAT_MASK_SID =

POINT_STAT_MESSAGE_TYPE = ADPSFC

# INFO TO PASS TO PYTHON SCRIPT
[user_env_vars]

AIRPORT = {custom?fmt=%s}
SOUNDING_FLAG = {PY_SOUNDING_FLAG}
PT_DELTA = {PY_PT_DELTA}
VAL_TIME = {PY_VAL_TIME}
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *PointStat MET Configuration* (page 218) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
```

(continues on next page)

(continued from previous page)

```
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses a Python embedding script to read input data

parm/use_cases/model_applications/pbl/PointStat_fcstHRRR_obsAMDAR_PBLH_PyEmbed/calc_amdar_pblh.py

```
'''
This script reads AMDAR hourly netcdf files, computes PBLH, and sends 11-column ascii table␣
↪to MET for point-stat
See accompanying PointStat_python_embedding_obs_amdar_pblh.conf for settings and passing in␣
↪env variables here
Jason M. English, May 2023
'''

import os
import sys
import pandas as pd
import numpy as np
import netCDF4 as nc

# silence this annoying warning about numpy bool being deprecated in favor of python bool
from warnings import filterwarnings
filterwarnings(action='ignore', category=DeprecationWarning, message='`np.bool` is a␣
↪deprecated alias')

#########################################################################

print("Python Script:\t" + repr(sys.argv[0]))

##
##  input file specified on the command line
##  load the data into the numpy array
##

loc_name = os.environ.get('AIRPORT') # "DENVER", "DALLAS", "BOSTON", "MINNEAPOLIS" # airport,
↪ not city
sf_include = os.environ.get('SOUNDING_FLAG')  # what sounding flags to include: ASC or DESC
pt_delta = float(os.environ.get('PT_DELTA')) #1.25  # potential temperature delta that␣
↪triggers PBLH calculation (K)
val_time = os.environ.get('VAL_TIME')  # valid time for this call (nearest hour)
rbox = 2.0   #  +/- deg;  set bigger than you need so MET mask can cut from that later
alt_base = 200.  # highest altitude where the alt min is looked for to get base potential␣
↪temperature
gap_max = 400.  # maximum allowable altitude gap (m) between the computed PBLH and the␣
↪altitude of the data point below it (gap_max + pblh/20)
alt_dp = 4    # minimum number of data points for the flight to be considered
alt_adj = "yes"  # adjust minimum altitude to be >= 0 yes or no
```

(continues on next page)

(continued from previous page)

```
out_rej = 2  # number of sigmas to trigger outlier reject

if loc_name == "DENVER":
   gnd0 = 5300.*0.3048   # surface elevation at location (msl);  DIA is at 5430ft but up to␣
↪300 feet lower than that within 500m of the airport
   lat0 = 39.856
   lon0 = -104.6764
elif loc_name == "DALLAS":
   gnd0 = 550.*0.3048
   lat0 = 32.8998
   lon0 = -97.0403
elif loc_name == "MINNEAPOLIS":
   gnd0 = 840.*0.3048
   lat0 = 44.8848
   lon0 = -93.2223
elif loc_name == "BOSTON":
   gnd0 = 20.*0.3048
   lat0 = 42.3656
   lon0 = -71.0096

# convert tail number array to readable character string
def get_tn(tn):
  tnc = tn.astype(str)   # convert to character string
  tnc = np.char.array(tnc)   # removes whitespace and allows vectorized string operations
  tnc_splice = tnc[:,0]+tnc[:,1]+tnc[:,2]+tnc[:,3]+tnc[:,4]+tnc[:,5]+tnc[:,6]+tnc[:,7]+tnc[:,
↪8]   # tail number spliced into a single string
  return tnc_splice

if len(sys.argv) != 2:
   print("ERROR: calc_amdar_pblh.py -> Must specify exactly one input file.")
   sys.exit(1)

# Read the input file as the first argument
input_file = os.path.expandvars(sys.argv[1])
try:
   print("Input File:\t" + repr(input_file))
   ncf = nc.Dataset(input_file)

   tn = ncf['tailNumber'][:]
   sf = ncf['sounding_flag'][:]  # -1=descent, 0=cruising, 1=ascent
   t = ncf['temperature'][:]
   alt = ncf['altitude'][:] - gnd0   # subtract surface height to get AGL
   lat = ncf['latitude'][:]
   lon = ncf['longitude'][:]
   ncf.close()
```

(continues on next page)

(continued from previous page)

```
  # set to NaN if cruising flight (not ascent or descent)
  t = np.where(sf == 0, np.nan, t)
  if sf_include == "ASC":
    t = np.where(sf == -1, np.nan, t)  # discard descents
  if sf_include == "DESC":
    t = np.where(sf == 1, np.nan, t)  # discard ascents

  # set to NaN if outside alt, lat/lon  bounds
  t = np.where((lat > lat0-rbox) & (lat < lat0+rbox) & (lon > lon0-rbox ) & (lon <␣
↪lon0+rbox), t, np.nan)

  # convert tail number array to readable character string
  tns = get_tn(tn)

  # set tail number array to NaN wherever temperature array is NaN
  tns = np.where(np.isnan(t), np.nan, tns)

  # get unique tail numbers within the specified lat/lon range
  tn_list = np.unique(tns)
  nflight = tn_list.size

  # Create arrays for saving PBLH and other fields for each flight
  pblh = np.full([nflight],np.nan)
  pblh_o = np.full([nflight],np.nan)   #pblh interpolated with outliers excluded
  pt_min = np.full([nflight],np.nan)  # potential temperature minimum
  lat_avg = np.full([nflight],np.nan)
  lon_avg = np.full([nflight],np.nan)

  for i,tn_name in enumerate(tn_list):  #loop through tail numbers

     if tn_name != "nan":
       tn_arr = np.where(tns == tn_name, tns, 'null')   # set array to null if it doesn't␣
↪this tail number
       tn_ind = np.where(tns == tn_arr)   # get list of indices

       # take the elements from each array ing only this flight (via the specified indices)
       tn_i  = np.squeeze(np.take(tn_arr, tn_ind))
       sf_i  = np.squeeze(np.take(sf, tn_ind))
       t_i   = np.squeeze(np.take(t, tn_ind))
       alt_i = np.squeeze(np.take(alt, tn_ind))
       lat_i = np.squeeze(np.take(lat, tn_ind))
       lon_i = np.squeeze(np.take(lon, tn_ind))

       # only include ascents/descents that have enough altitude/temperature data
```

(continues on next page)

(continued from previous page)

```
        if (np.amin(alt_i) < alt_base) & (np.amax(alt_i) > alt_base) & (alt_i.size >= alt_
↪dp):

          # sort altitude and temperature arrays to be ascending
          sort_inds = np.argsort(alt_i)
          t_d = np.copy(t_i[sort_inds])
          alt_d = np.copy(alt_i[sort_inds])
          lat_d = np.copy(lat_i[sort_inds])
          lon_d = np.copy(lon_i[sort_inds])

          # adjust altitude minimum to zero if it's negative
          if alt_adj == "yes":
            if np.nanmin(alt_d) < 0:
              alt_d[:] = alt_d[:] - np.nanmin(alt_d)

          # convert altitude to pressure
          slp = 101325.  # Sea level pressure (Pa)
          expon = (-9.80665 *0.0289644) / (8.31432 * -0.0065)
          p_d = slp * (1. - (alt_d + gnd0)/44307.694)**expon   # needs to be pressure␣
↪altitude (add ground ht)

          # convert temperature to potential temperature
          pt_d = np.copy(t_d)
          pt_d[:] = t_d[:] * (slp/p_d[:])**0.286

          # Find minimum potential temperature that occurs below the specified altitude alt_
↪base
          pt_min[i] = np.nanmin(np.where(alt_d < alt_base, pt_d, np.nan))
          pt_min_ind = np.where(pt_d == pt_min[i])[0][0]  # find array indexing that value

          # Only move forward if minimum PT is within a reasonable range
          if (pt_min[i] > 0) & (pt_min[i] < 3040):

          # consider only potential temperature values above where pt_min was found when␣
↪searching for pblh
            alt_d[:pt_min_ind] = np.nan
            pt_d[:pt_min_ind] = np.nan
            pt_dif = np.copy(pt_d)
            pt_dif[:] = pt_d[:] - pt_min[i]

            # determine lowest height that exceeds the specified pt_delta (K)
            if np.nanmax(pt_d) >= (pt_min[i]+pt_delta):    # make sure it exists in this␣
↪profile
              pblh_alt = np.nanmin(np.where(pt_d >= (pt_min[i]+pt_delta),alt_d, np.nan))
              pblh_ind = np.where(alt_d == pblh_alt)[0][0]   # altitude index where pblh is␣
```

(continues on next page)

(continued from previous page)

```
↪found

            # only include pblh if the altitude below it isn't too big of a gap
            if pblh_ind.size == 1:   # make sure only 1 index was found
              alt_gap = alt_d[pblh_ind]-alt_d[pblh_ind-1]

              if alt_gap < (gap_max + alt_d[pblh_ind]/20.):
                pblh[i] = alt_d[pblh_ind]

                # linear interpolate PBLH between this data point and the one below it
                pblh[i] = np.interp((pt_min[i]+pt_delta), pt_d[pblh_ind-1:pblh_ind+1], alt_
↪d[pblh_ind-1:pblh_ind+1])

                # compute lat/lon for this flight by taking the avg lat/lon coordiantes␣
↪over the flight
                lat_avg[i] = np.average(lat_i)
                lon_avg[i] = np.average(lon_i)

         print("tn=", tn_i[0], ", sf=", sf_include, ", n=", pt_d.size, ", pt_min (K)=", np.
↪array2string(pt_min[i]),
               ", pblh interp (m)=", np.array2string(pblh[i]), ", pblh closest (m)=", np.
↪array2string(pblh[i]))

#########################################################################

  # Now that all flights at this hour have been computed, conduct statistics and averaging␣
↪on them
  if np.count_nonzero(~np.isnan(pblh)) > 0:
    pblh_o[:] = np.where((pblh >= np.nanmean(pblh)-float(out_rej)*np.nanstd(pblh)) &
                         (pblh <= np.nanmean(pblh)+float(out_rej)*np.nanstd(pblh)), pblh,␣
↪np.nan)

    # only include flights with a computed PBLH value
    lat_avg[np.isnan(pblh_o)] = np.nan
    lon_avg[np.isnan(pblh_o)] = np.nan
    pblh_o = pblh_o[~np.isnan(pblh_o)]
    lat_avg= lat_avg[~np.isnan(lat_avg)]
    lon_avg= lon_avg[~np.isnan(lon_avg)]

  # Read and format the input 11-column observations:
  #   (1)  string:  Message_Type ('ADPSFC')
  #   (2)  string:  Station_ID (AIRPORT)
  #   (3)  string:  Valid_Time(YYYYMMDD_HHMMSS)
  #   (4)  numeric: Lat(Deg North)
  #   (5)  numeric: Lon(Deg East)
```

(continues on next page)

(continued from previous page)

```
   #   (6)  numeric: Elevation(msl)
   #   (7)  string:  Var_Name(or GRIB_Code)
   #   (8)  numeric: Level
   #   (9)  numeric: Height(msl or agl)
   #   (10) string:  QC_String
   #   (11) numeric: Observation_Value

   point_data = pd.DataFrame({'typ':'ADPSFC', 'sid':loc_name, 'vld':val_time,
                              'lat':lat_avg, 'lon':lon_avg, 'elv':gnd0, 'var':'HPBL',
                              'lvl':0, 'hgt':0, 'qc':'AMDAR', 'obs':pblh_o}).values.tolist()

   print(point_data)
   print("     point_data: Data Length:\t" + repr(len(point_data)))
   print("     point_data: Data Type:\t" + repr(type(point_data)))

except NameError:
    print("Can't find the input file")
    sys.exit(1)
```

Running METplus

It is recommended to run this use case by:

Passing in PointStat_fcstHRRR_obsAMDAR_PBLH_PyEmbed.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/PointStat_fcstHRRR_obsAMDAR_PBLH_PyEmbed.conf -c /path/to/user_
↪system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in point_stat_pblh (relative to **OUTPUT_BASE**) with subdirectories for valid time (YYYYMMDD) and will contain .stat files with the following naming convention:

convention:
point_stat_{AIRPORT}_{SOUNDING_FLAG}_{PT_DELTA}_{LEADTIME}L_{VALIDTIME}.stat
example: point_stat_DENVER_ALL_1.25_120000L_20220701_200000V.stat

Keywords

Note:

- GenVxMaskToolUseCase
- PointStatToolUseCase
- PythonEmbeddingFileUseCase
- PBLAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/pbl-PointStat_fcstHRRR_obsAMDAR_PBLH_PyEmbed.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.9 Precipitation

Any fields that can be defined as precipitation, including rain, snow, and other precipitation types

7.2.16.9.1 Gen-Ens-Prod: Basic Post-Processing only

model_application/precipitation/GenEnsProd_fcstHRRRE _FcstOnly_NetCDF.conf

Scientific Objective

Post-process ensemble members to derive simple (non-bias-corrected) mean, standard deviation (spread), minimum, maximum, and range fields for use in other MET tools.

Datasets

- Forecast dataset: HRRRE 3 member ensemble netcdf 3 hour precipitation accumulation

METplus Components

This use case runs gen_ens_prod on HRRRE data from 3 members after running it through pcp_combine to create a 3 hour precipitation accumulation

METplus Workflow

The following tools are used for each run time: GenEnsProd

This example loops by initialization time. For each initialization time it will process forecast leads 3, 6, 9 and 12

Run times:

Init: 2019-05-19_12Z
Forecast lead: 3

Init: 2019-05-19_12Z
Forecast lead: 6

Init: 2019-05-19_12Z
Forecast lead: 9

Init: 2019-05-19_12Z
Forecast lead: 12

Init: 2019-05-20_00Z
Forecast lead: 3

Init: 2019-05-20_00Z

Forecast lead: 6

Init: 2019-05-20_00Z
Forecast lead: 9

Init: 2019-05-20_00Z
Forecast lead: 12

METplus Configuration

METplus first loads all of the configurations found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/precipitation/
↪GenEnsProd_fcstHRRRE_FcstOnly_NetCDF.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GenEnsProd


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###
```

(continues on next page)

(continued from previous page)

```
LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2019051912
INIT_END=2019052000
INIT_INCREMENT=43200

LEAD_SEQ = 3,6,9,12


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

GEN_ENS_PROD_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/HRRRE/pcp_combine
GEN_ENS_PROD_INPUT_TEMPLATE = hrrrebegin_end_incr(1,3,1,2)_{init?fmt=%Y%m%d%H}f{lead?fmt=
↪%HHH}_A03.nc

GEN_ENS_PROD_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GenEnsProd_
↪fcstHRRRE_FcstOnly_NetCDF/GenEnsProd
GEN_ENS_PROD_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H%M}/gen_ens_prod_{ENS_VAR1_NAME}_{valid?fmt=
↪%Y%m%d_%H%M%S}V_ens.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

ENS_VAR1_NAME = APCP_03
ENS_VAR1_LEVELS = "(*,*)"


###
# GenEnsProd Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#genensprod
###

GEN_ENS_PROD_N_MEMBERS = 3

GEN_ENS_PROD_ENS_THRESH = 0.5
GEN_ENS_PROD_VLD_THRESH = 1.0

GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON = TRUE
```

(continues on next page)

(continued from previous page)

```
GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_MINUS = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_PLUS = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_MIN = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_MAX = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_RANGE = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_VLD_COUNT = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_NEP = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_NMEP = FALSE
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GenEnsProd MET Configuration* (page 121) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Gen-Ens-Prod configuration file.
//
// For additional information, please see the MET Users Guide.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
```

(continues on next page)

(continued from previous page)

```
//desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
//censor_thresh =
${METPLUS_CENSOR_THRESH}

//censor_val    =
${METPLUS_CENSOR_VAL}

//normalize =
${METPLUS_NORMALIZE}

//cat_thresh    =
${METPLUS_CAT_THRESH}

//nc_var_str    =
${METPLUS_NC_VAR_STR}

//
// Ensemble fields to be processed
//
ens = {
   //file_type =
   ${METPLUS_ENS_FILE_TYPE}

   //ens_thresh =
   ${METPLUS_ENS_THRESH}

   //vld_thresh =
   ${METPLUS_VLD_THRESH}
```

(continues on next page)

(continued from previous page)

```
   //field =
   ${METPLUS_ENS_FIELD}

}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}


////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood ensemble probabilities
//
//nbrhd_prob = {
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
//nmep_smooth = {
${METPLUS_NMEP_SMOOTH_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}

//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Ensemble product output types
// May be set separately in each "ens.field" entry
//
//ensemble_flag = {
${METPLUS_ENSEMBLE_FLAG_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//version = "V10.1.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

The command to run this use case is:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/precipitation/GenEnsProd_
↪fcstHRRRE_FcstOnly_NetCDF.conf
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/precipitation/GenEnsProd_fcstHRRRE_FcstOnly_NetCDF/GenEnsProd (relative to **OUTPUT_BASE**) The following folder/file combination will be created:

-201905191200

- gen_ens_prod_APCP_03_20190519_150000V_ens.nc
- gen_ens_prod_APCP_03_20190519_180000V_ens.nc
- gen_ens_prod_APCP_03_20190519_210000V_ens.nc
- gen_ens_prod_APCP_03_20190520_000000V_ens.nc

-201905200000

- gen_ens_prod_APCP_03_20190520_030000V_ens.nc
- gen_ens_prod_APCP_03_20190520_060000V_ens.nc
- gen_ens_prod_APCP_03_20190520_090000V_ens.nc
- gen_ens_prod_APCP_03_20190520_120000V_ens.nc

Keywords

Note:

- GenEnsProdToolUseCase
- NOAAHWTOrgUseCase
- PrecipitationAppUseCase
- NetCDFFileUseCase
- EnsembleAppUseCase
- ConvectionAllowingModelsAppUseCase
- ProbabilityGenerationAppUseCase
- ListExpansionFeatureUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/precipitation-GenEnsProd_fcstHRRRE_FcstOnly_NetCDF.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.9.2 Grid-Stat: 6hr QPF in NetCDF format

model_applications/precipitation/GridStat_fcstHREFmean _obsStgIV_Netcdf.conf

Scientific Objective

Evaluate the skill of a high resolution multi-model ensemble mean at predicting 6 hour precipitation accumulation using the NCEP Stage IV gauge corrected analysis.

Datasets

Describe the datasets here. Relevant information about the datasets that would be beneficial include:

- Forecast dataset: HREF mean forecasts in NetCDF
- Observation dataset: Stage IV GRIB 6 hour precipitation accumulation

METplus Components

This use case first runs PCPCombine on the forecast data to build a 6 hour precipitation accumulation from 1 hour files or a single 6 hour file. Then the observation data is regridded to the model grid using the RegridDataPlane. Finally, the observation files are compared to the forecast data using GridStat.

METplus Workflow

The following tools are used for each run time: PCPCombine (observation) > RegridDataPlane (observation) > GridStat

This example loops by initialization time. There is only one initalization time in this example so the following will be run:

Run times:

Init: 2017-05-09_12Z
Forecast lead: 18

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/precipitation/GridStat_fcstHREFmean_obsStgIV_NetCDF.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/precipitation/
↪GridStat_fcstHREFmean_obsStgIV_NetCDF.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PCPCombine, RegridDataPlane, GridStat


###
```

(continues on next page)

(continued from previous page)

```
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2017050912
INIT_END=2017050912
INIT_INCREMENT=43200

LEAD_SEQ = 18


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_PCP_COMBINE_RUN = True

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/HREFv2_Mean
FCST_PCP_COMBINE_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/hrefmean_{init?fmt=%Y%m%d%H}f{lead?fmt=
↪%HHH}.nc

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_
↪fcstHREFmean_obsStgIV_NetCDF/HREFv2_Mean/bucket
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/hrefmean_{valid?fmt=%Y%m%d%H}_A{level?
↪fmt=%HH}.nc


OBS_REGRID_DATA_PLANE_RUN = True

OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/StageIV
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = {valid?fmt=%Y%m%d?shift=-12H}12_st4.nc

OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_
↪fcstHREFmean_obsStgIV_NetCDF/StageIV_netcdf/regrid
```

(continues on next page)

(continued from previous page)

```
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}_st4_A06.nc


FCST_GRID_STAT_INPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
FCST_GRID_STAT_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/hrefmean_{valid?fmt=%Y%m%d%H}_A06.nc

OBS_GRID_STAT_INPUT_DIR = {OBS_REGRID_DATA_PLANE_OUTPUT_DIR}
OBS_GRID_STAT_INPUT_TEMPLATE = {OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE}

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_fcstHREFmean_
↪obsStgIV_NetCDF/GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H%M}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = HREF_MEAN
OBTYPE = STAGE4

FCST_VAR1_NAME = {FCST_PCP_COMBINE_OUTPUT_NAME}
FCST_VAR1_LEVELS = "(*,*)"
FCST_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2, gt152.4

OBS_VAR1_NAME = {OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME}
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2, gt152.4


OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = P06M_NONE
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = "({valid?fmt=%Y%m%d_%H%M%S},*,*)"

OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = P06M_NONE

###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

FCST_PCP_COMBINE_METHOD = ADD

FCST_PCP_COMBINE_INPUT_DATATYPE = NETCDF
```

(continues on next page)

(continued from previous page)

```
FCST_PCP_COMBINE_CONSTANT_INIT = true

FCST_PCP_COMBINE_INPUT_ACCUMS = 1
FCST_PCP_COMBINE_INPUT_NAMES = P01M_NONE
FCST_PCP_COMBINE_INPUT_LEVELS = "(0,*,*)"

FCST_PCP_COMBINE_OUTPUT_ACCUM = 6
FCST_PCP_COMBINE_OUTPUT_NAME = APCP_06

OBS_PCP_COMBINE_INPUT_DATATYPE = NETCDF


###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

REGRID_DATA_PLANE_METHOD = BUDGET
REGRID_DATA_PLANE_WIDTH = 2

REGRID_DATA_PLANE_VERIF_GRID={INPUT_BASE}/model_applications/precipitation/mask/CONUS_
↪HRRRTLE.nc


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_REGRID_TO_GRID = OBS

GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 7, 15
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_DMAP = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE
GRID_STAT_NC_PAIRS_FLAG_DISTANCE_MAP = TRUE
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
```

(continues on next page)

(continued from previous page)

```
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
```

(continues on next page)

(continued from previous page)

```
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstHREFmean_obsStgIV_NetCDF.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪GridStat_fcstHREFmean_obsStgIV_NetCDF.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstHREFmean_obsStgIV_NetCDF.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪GridStat_fcstHREFmean_obsStgIV_NetCDF.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify

the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/precipitation/GridStat_fcstHREFmean_obsStgIV_NetCDF/GridStat/201705091200 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_000000L_20170510_060000V_pairs.nc
- grid_stat_000000L_20170510_060000V.stat

Keywords

Note:

- GridStatToolUseCase
- PrecipitationAppUseCase
- PCPCombineToolUseCase
- RegridDataPlaneToolUseCase
- NetCDFFileUseCase
- NOAAWPCOrgUseCase
- NOAAHMTOrgUseCase

- NOAAHWTOrgUseCase
- ConvectionAllowingModelsAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/precipitation-GridStat_fcstHREFmean_obsStgIV_NetCDF.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.9.3 Ensemble-Stat: WoFS

model_application/precipitation/EnsembleStat_fcstWOFS_obsWOFS.conf

Scientific Objective

Comparing the Warn on Forecast System (WoFS) ensemble to the MRMS observed variable field to understand its forecasting abilities. Specifically focusing on accumulated precipitation at different neighborhood distances and accumulation thresholds to provide meaningful analysis output that can provide direction to future WoFS improvement.

Datasets

- Forecast dataset: WoFS Ensemble

METplus Components

This use case runs PCP-Combine on each ensemble member, then runs Ensemble-Stat on the output. Finally, it runs Grid-Stat on the output from Ensemble-Stat

METplus Workflow

The following tools are used for each run time: PCPCombine, EnsembleStat, GridStat

This example loops by initialization time. For each initialization time it will process the 1 hour forecast lead

Run times:

Init: 2020-06-15_17Z
Forecast lead: 1 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/precipitation/EnsembleStat_fcstWOFS_obsWOFS.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/precipitation/
↪EnsembleStat_fcstWOFS_obsWOFS.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PcpCombine, EnsembleStat, GenEnsProd, GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H%M
INIT_BEG = 202006151700
INIT_END = 202006151700
INIT_INCREMENT = 3600

LEAD_SEQ = 1


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

# PCPCombine

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/WOFS/ensemble
FCST_PCP_COMBINE_INPUT_TEMPLATE = {init?fmt=%Y%m%d?shift=-44100}/{init?fmt=%H%M}/ENS_MEM_
↪{custom?fmt=%s}

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/{MODEL}/pcp_combine
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d?shift=-44100}/{init?fmt=%H%M}/ENS_MEM_
↪{custom?fmt=%s}/wofs{custom?fmt=%s}_PCP_{init?fmt=%Y%m%d}_{init?fmt=%H%M}_{valid?fmt=%H%M}_
↪A1.nc

# GenEnsProd

GEN_ENS_PROD_INPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
GEN_ENS_PROD_INPUT_TEMPLATE = {init?fmt=%Y%m%d?shift=-44100}/{init?fmt=%H%M}/ENS_MEM_??/wofs?
↪?_PCP_{init?fmt=%Y%m%d}_{init?fmt=%H%M}_{valid?fmt=%H%M}_A1.nc

GEN_ENS_PROD_OUTPUT_DIR = {OUTPUT_BASE}/{MODEL}/ensemble_stat
GEN_ENS_PROD_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d?shift=-44100}/{init?fmt=%H%M}/gen_ens_prod_
↪{MODEL}_PCP_{init?fmt=%H%M}_{lead?fmt=%H%M}00L_A1_{init?fmt=%Y%m%d}_{valid?fmt=%H%M}00V_
↪ens.nc

# EnsembleStat

FCST_ENSEMBLE_STAT_INPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d?shift=-44100}/{init?fmt=%H%M}/ENS_MEM_??
↪/wofs??_PCP_{init?fmt=%Y%m%d}_{init?fmt=%H%M}_{valid?fmt=%H%M}_A1.nc

OBS_ENSEMBLE_STAT_GRID_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/WOFS/OBS
OBS_ENSEMBLE_STAT_GRID_INPUT_TEMPLATE = {init?fmt=%Y%m%d?shift=-44100}/{init?fmt=%H%M}/mrms_
↪PCP_{init?fmt=%Y%m%d}_{init?fmt=%H%M}_{valid?fmt=%H%M}_A1.nc

ENSEMBLE_STAT_OUTPUT_DIR = {OUTPUT_BASE}/{MODEL}/ensemble_stat
ENSEMBLE_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d?shift=-44100}/{init?fmt=%H%M}

# GridStat

FCST_GRID_STAT_INPUT_DIR = {GEN_ENS_PROD_OUTPUT_DIR}
FCST_GRID_STAT_INPUT_TEMPLATE = {GEN_ENS_PROD_OUTPUT_TEMPLATE}

OBS_GRID_STAT_INPUT_DIR = {OBS_ENSEMBLE_STAT_GRID_INPUT_DIR}
OBS_GRID_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d?shift=-44100}/{init?fmt=%H%M}/mrms_PCP_{init?
```

(continues on next page)

(continued from previous page)

```
↪fmt=%Y%m%d}_{init?fmt=%H%M}_{valid?fmt=%H%M}_A1.nc

GRID_STAT_VERIFICATION_MASK_TEMPLATE = {INPUT_BASE}/model_applications/precipitation/WOFS/
↪domain/WOFS_domain_{init?fmt=%Y%m%d}.nc

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/{MODEL}/grid_stat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d?shift=-44100}/{init?fmt=%H%M}/


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL= WOFS
OBTYPE = MRMS_QPE


# GenEnsProd

ENS_VAR1_NAME = APCP_01
ENS_VAR1_LEVELS = "(*,*)"
ENS_VAR1_THRESH = >=12.7, >=25.4, >=50.8


# EnsembleStat

FCST_ENSEMBLE_STAT_VAR1_NAME = APCP_01
FCST_ENSEMBLE_STAT_VAR1_LEVELS = "(*,*)"
FCST_ENSEMBLE_STAT_VAR1_THRESH = >=12.7, >=25.4, >=50.8

OBS_ENSEMBLE_STAT_VAR1_NAME = GaugeCorrQPE01H_01
OBS_ENSEMBLE_STAT_VAR1_LEVELS = "(*,*)"
OBS_ENSEMBLE_STAT_VAR1_THRESH = {FCST_ENSEMBLE_STAT_VAR1_THRESH}


# GridStat

FCST_GRID_STAT_VAR1_NAME = APCP_01_A1_ENS_NMEP_ge12.7_NBRHD1_NEAREST1
FCST_GRID_STAT_VAR1_LEVELS = "(*,*)"
FCST_GRID_STAT_VAR1_OPTIONS = prob = TRUE
FCST_GRID_STAT_VAR1_THRESH = >=0.0, >=0.05, >=0.15, >=0.25, >=0.35, >=0.45, >=0.55, >=0.65, >
↪=0.75, >=0.85, >=0.95, >=1.0
#
FCST_GRID_STAT_VAR2_NAME = APCP_01_A1_ENS_NMEP_ge25.4_NBRHD1_NEAREST1
```

(continues on next page)

(continued from previous page)

```
FCST_GRID_STAT_VAR2_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR2_OPTIONS = {FCST_GRID_STAT_VAR1_OPTIONS}
FCST_GRID_STAT_VAR2_THRESH = {FCST_GRID_STAT_VAR1_THRESH}

FCST_GRID_STAT_VAR3_NAME = APCP_01_A1_ENS_NMEP_ge50.8_NBRHD1_NEAREST1
FCST_GRID_STAT_VAR3_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR3_OPTIONS = {FCST_GRID_STAT_VAR1_OPTIONS}
FCST_GRID_STAT_VAR3_THRESH = {FCST_GRID_STAT_VAR1_THRESH}

FCST_GRID_STAT_VAR4_NAME = APCP_01_A1_ENS_NMEP_ge12.7_NBRHD25_NEAREST1
FCST_GRID_STAT_VAR4_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR4_OPTIONS = {FCST_GRID_STAT_VAR1_OPTIONS}
FCST_GRID_STAT_VAR4_THRESH = {FCST_GRID_STAT_VAR1_THRESH}

FCST_GRID_STAT_VAR5_NAME = APCP_01_A1_ENS_NMEP_ge25.4_NBRHD25_NEAREST1
FCST_GRID_STAT_VAR5_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR5_OPTIONS = {FCST_GRID_STAT_VAR1_OPTIONS}
FCST_GRID_STAT_VAR5_THRESH = {FCST_GRID_STAT_VAR1_THRESH}

FCST_GRID_STAT_VAR6_NAME = APCP_01_A1_ENS_NMEP_ge50.8_NBRHD25_NEAREST1
FCST_GRID_STAT_VAR6_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR6_OPTIONS = {FCST_GRID_STAT_VAR1_OPTIONS}
FCST_GRID_STAT_VAR6_THRESH = {FCST_GRID_STAT_VAR1_THRESH}

FCST_GRID_STAT_VAR7_NAME = APCP_01_A1_ENS_NMEP_ge12.7_NBRHD81_NEAREST1
FCST_GRID_STAT_VAR7_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR7_OPTIONS = {FCST_GRID_STAT_VAR1_OPTIONS}
FCST_GRID_STAT_VAR7_THRESH = {FCST_GRID_STAT_VAR1_THRESH}

FCST_GRID_STAT_VAR8_NAME = APCP_01_A1_ENS_NMEP_ge25.4_NBRHD81_NEAREST1
FCST_GRID_STAT_VAR8_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR8_OPTIONS = {FCST_GRID_STAT_VAR1_OPTIONS}
FCST_GRID_STAT_VAR8_THRESH = {FCST_GRID_STAT_VAR1_THRESH}

FCST_GRID_STAT_VAR9_NAME = APCP_01_A1_ENS_NMEP_ge50.8_NBRHD81_NEAREST1
FCST_GRID_STAT_VAR9_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR9_OPTIONS = {FCST_GRID_STAT_VAR1_OPTIONS}
FCST_GRID_STAT_VAR9_THRESH = {FCST_GRID_STAT_VAR1_THRESH}

FCST_GRID_STAT_VAR10_NAME = APCP_01_A1_ENS_NMEP_ge12.7_NBRHD1_NEAREST1
FCST_GRID_STAT_VAR10_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR10_OPTIONS = prob = FALSE
FCST_GRID_STAT_VAR10_THRESH = >=0.5

FCST_GRID_STAT_VAR11_NAME = APCP_01_A1_ENS_NMEP_ge25.4_NBRHD1_NEAREST1
```

(continues on next page)

(continued from previous page)

```
FCST_GRID_STAT_VAR11_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR11_OPTIONS = {FCST_GRID_STAT_VAR10_OPTIONS}
FCST_GRID_STAT_VAR11_THRESH = {FCST_GRID_STAT_VAR10_THRESH}

FCST_GRID_STAT_VAR12_NAME = APCP_01_A1_ENS_NMEP_ge50.8_NBRHD1_NEAREST1
FCST_GRID_STAT_VAR12_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR12_OPTIONS = {FCST_GRID_STAT_VAR10_OPTIONS}
FCST_GRID_STAT_VAR12_THRESH = {FCST_GRID_STAT_VAR10_THRESH}

FCST_GRID_STAT_VAR13_NAME = APCP_01_A1_ENS_NMEP_ge12.7_NBRHD25_NEAREST1
FCST_GRID_STAT_VAR13_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR13_OPTIONS = {FCST_GRID_STAT_VAR10_OPTIONS}
FCST_GRID_STAT_VAR13_THRESH = {FCST_GRID_STAT_VAR10_THRESH}

FCST_GRID_STAT_VAR14_NAME = APCP_01_A1_ENS_NMEP_ge25.4_NBRHD25_NEAREST1
FCST_GRID_STAT_VAR14_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR14_OPTIONS = {FCST_GRID_STAT_VAR10_OPTIONS}
FCST_GRID_STAT_VAR14_THRESH = {FCST_GRID_STAT_VAR10_THRESH}

FCST_GRID_STAT_VAR15_NAME = APCP_01_A1_ENS_NMEP_ge50.8_NBRHD25_NEAREST1
FCST_GRID_STAT_VAR15_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR15_OPTIONS = {FCST_GRID_STAT_VAR10_OPTIONS}
FCST_GRID_STAT_VAR15_THRESH = {FCST_GRID_STAT_VAR10_THRESH}

FCST_GRID_STAT_VAR16_NAME = APCP_01_A1_ENS_NMEP_ge12.7_NBRHD81_NEAREST1
FCST_GRID_STAT_VAR16_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR16_OPTIONS = {FCST_GRID_STAT_VAR10_OPTIONS}
FCST_GRID_STAT_VAR16_THRESH = {FCST_GRID_STAT_VAR10_THRESH}

FCST_GRID_STAT_VAR17_NAME = APCP_01_A1_ENS_NMEP_ge25.4_NBRHD81_NEAREST1
FCST_GRID_STAT_VAR17_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR17_OPTIONS = {FCST_GRID_STAT_VAR10_OPTIONS}
FCST_GRID_STAT_VAR17_THRESH = {FCST_GRID_STAT_VAR10_THRESH}

FCST_GRID_STAT_VAR18_NAME = APCP_01_A1_ENS_NMEP_ge50.8_NBRHD81_NEAREST1
FCST_GRID_STAT_VAR18_LEVELS = {FCST_GRID_STAT_VAR1_LEVELS}
FCST_GRID_STAT_VAR18_OPTIONS = {FCST_GRID_STAT_VAR10_OPTIONS}
FCST_GRID_STAT_VAR18_THRESH = {FCST_GRID_STAT_VAR10_THRESH}

OBS_GRID_STAT_VAR1_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR1_LEVELS = "(*,*)"
OBS_GRID_STAT_VAR1_THRESH = >=12.7

OBS_GRID_STAT_VAR2_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR2_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
```

(continues on next page)

(continued from previous page)

```
OBS_GRID_STAT_VAR2_THRESH = >=25.4

OBS_GRID_STAT_VAR3_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR3_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR3_THRESH = >=50.8

OBS_GRID_STAT_VAR4_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR4_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR4_THRESH = >=12.7

OBS_GRID_STAT_VAR5_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR5_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR5_THRESH = >=25.4

OBS_GRID_STAT_VAR6_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR6_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR6_THRESH = >=50.8

OBS_GRID_STAT_VAR7_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR7_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR7_THRESH = >=12.7

OBS_GRID_STAT_VAR8_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR8_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR8_THRESH = >=25.4

OBS_GRID_STAT_VAR9_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR9_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR9_THRESH = >=50.8

OBS_GRID_STAT_VAR10_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR10_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR10_THRESH = >=12.7

OBS_GRID_STAT_VAR11_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR11_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR11_THRESH = >=25.4

OBS_GRID_STAT_VAR12_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR12_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR12_THRESH = >=50.8

OBS_GRID_STAT_VAR13_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR13_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR13_THRESH = >=12.7
```

(continues on next page)

(continued from previous page)

```
OBS_GRID_STAT_VAR14_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR14_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR14_THRESH = >=25.4

OBS_GRID_STAT_VAR15_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR15_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR15_THRESH = >=50.8

OBS_GRID_STAT_VAR16_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR16_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR16_THRESH = >=12.7

OBS_GRID_STAT_VAR17_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR17_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR17_THRESH = >=25.4

OBS_GRID_STAT_VAR18_NAME = GaugeCorrQPE01H_01
OBS_GRID_STAT_VAR18_LEVELS = {OBS_GRID_STAT_VAR1_LEVELS}
OBS_GRID_STAT_VAR18_THRESH = >=50.8


###
# PcpCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

FCST_PCP_COMBINE_RUN = TRUE
FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB
FCST_PCP_COMBINE_METHOD = USER_DEFINED
FCST_PCP_COMBINE_BUCKET_INTERVAL = 1
FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB
FCST_PCP_COMBINE_FIELD_NAME = APCP
FCST_PCP_COMBINE_INPUT_ACCUMS = 1

PCP_COMBINE_SKIP_IF_OUTPUT_EXISTS = TRUE

PCP_COMBINE_CUSTOM_LOOP_LIST = 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15,␣
↪16, 17, 18

FCST_PCP_COMBINE_COMMAND = -sum 00000000_000000 1 {valid?fmt=%Y%m%d}_{valid?fmt=%H%M}00 1 -
↪pcpdir {FCST_PCP_COMBINE_INPUT_DIR}/{FCST_PCP_COMBINE_INPUT_TEMPLATE} -pcprx wofs -field
↪'name="APCP";level="A1";'
```

(continues on next page)

(continued from previous page)

```
###
# GenEnsProd Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#genensprod
###


GEN_ENS_PROD_N_MEMBERS = 18

GEN_ENS_PROD_ENS_THRESH = 1.0

GEN_ENS_PROD_REGRID_TO_GRID = FCST
GEN_ENS_PROD_REGRID_METHOD = BUDGET
GEN_ENS_PROD_REGRID_WIDTH = 2
GEN_ENS_PROD_REGRID_VLD_THRESH = 1.0

GEN_ENS_PROD_NBRHD_PROB_WIDTH = 1, 3, 5, 7, 9
GEN_ENS_PROD_NBRHD_PROB_SHAPE = SQUARE
GEN_ENS_PROD_NBRHD_PROB_VLD_THRESH = 1.0

GEN_ENS_PROD_NMEP_SMOOTH_VLD_THRESH = 1.0
GEN_ENS_PROD_NMEP_SMOOTH_SHAPE = SQUARE
GEN_ENS_PROD_NMEP_SMOOTH_METHOD = NEAREST
GEN_ENS_PROD_NMEP_SMOOTH_WIDTH = 1

GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_MINUS = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_PLUS = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_MIN = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_MAX = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_RANGE = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_VLD_COUNT = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_NEP = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_NMEP = TRUE

###
# EnsembleStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ensemblestat
###

ENSEMBLE_STAT_N_MEMBERS = 18

ENSEMBLE_STAT_ENS_THRESH = 1.0
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_REGRID_TO_GRID = FCST
ENSEMBLE_STAT_REGRID_METHOD = BUDGET
ENSEMBLE_STAT_REGRID_WIDTH = 2
ENSEMBLE_STAT_REGRID_VLD_THRESH = 1.0

FCST_ENSEMBLE_STAT_INPUT_GRID_DATATYPE = NETCDF
OBS_ENSEMBLE_STAT_INPUT_GRID_DATATYPE = NETCDF


ENSEMBLE_STAT_MESSAGE_TYPE =

ENSEMBLE_STAT_OUTPUT_FLAG_ECNT = STAT
ENSEMBLE_STAT_OUTPUT_FLAG_RPS = NONE
ENSEMBLE_STAT_OUTPUT_FLAG_RHIST = STAT
ENSEMBLE_STAT_OUTPUT_FLAG_PHIST = STAT
ENSEMBLE_STAT_OUTPUT_FLAG_ORANK = STAT
ENSEMBLE_STAT_OUTPUT_FLAG_SSVAR = STAT
ENSEMBLE_STAT_OUTPUT_FLAG_RELP = STAT

ENSEMBLE_STAT_NC_ORANK_FLAG_LATLON = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_MEAN = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_RAW = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_RANK = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_PIT = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_VLD_COUNT = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_WEIGHT = FALSE

ENSEMBLE_STAT_OUTPUT_PREFIX = {MODEL}_PCP_{init?fmt=%H%M}_{lead?fmt=%H%M}00L_A1


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_REGRID_TO_GRID = FCST

FCST_GRID_STAT_INPUT_GRID_DATATYPE = NETCDF
OBS_GRID_STAT_INPUT_GRID_DATATYPE = NETCDF
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE
GRID_STAT_NEIGHBORHOOD_WIDTH = 1, 3, 5

GRID_STAT_OUTPUT_PREFIX = {MODEL}_PCP_{init?fmt=%H%M}_A1
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

EnsembleStatConfig_wrapped

Note: See the *EnsembleStat MET Configuration* (page 105) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Ensemble-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
${METPLUS_DESC}

//
// Output observation type to be written
//
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
${METPLUS_REGRID_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
${METPLUS_CENSOR_THRESH}
${METPLUS_CENSOR_VAL}
cat_thresh    = [];
nc_var_str    = "";

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}


////////////////////////////////////////////////////////////////////////////////

//prob_cat_thresh =
${METPLUS_PROB_CAT_THRESH}

//prob_pct_thresh =
${METPLUS_PROB_PCT_THRESH}

//eclv_points =
${METPLUS_ECLV_POINTS}


////////////////////////////////////////////////////////////////////////////////

//
// Forecast and observation fields to be verified
//

fcst = {

   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_ENS_THRESH}
   ${METPLUS_VLD_THRESH}
   ${METPLUS_FCST_FIELD}
}

obs = {
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//

${METPLUS_MESSAGE_TYPE}
sid_exc        = [];
//obs_thresh     = [ NA ];
${METPLUS_OBS_THRESH}

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

${METPLUS_DUPLICATE_FLAG}
obs_summary    = NONE;
obs_perc_value = 50;
${METPLUS_SKIP_CONST}

//
// Observation error options
// Set dist_type to NONE to use the observation error table instead
// May be set separately in each "obs.field" entry
//
obs_error = {
   ${METPLUS_OBS_ERROR_FLAG}
   dist_type        = NONE;
   dist_parm        = [];
   inst_bias_scale  = 1.0;
   inst_bias_offset = 0.0;
   min              = NA;      // Valid range of data
   max              = NA;
}

//
// Mapping of message type group name to comma-separated list of values.
```

(continues on next page)

(continued from previous page)

```
//
message_type_group_map = [
   { key = "SURFACE"; val = "ADPSFC,SFCSHP,MSONET";               },
   { key = "ANYAIR";  val = "AIRCAR,AIRCFT";                      },
   { key = "ANYSFC";  val = "ADPSFC,SFCSHP,ADPUPA,PROFLR,MSONET"; },
   { key = "ONLYSF";  val = "ADPSFC,SFCSHP";                      }
];

//
// Ensemble bin sizes
// May be set separately in each "obs.field" entry
//
${METPLUS_ENS_SSVAR_BIN_SIZE}
${METPLUS_ENS_PHIST_BIN_SIZE}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}



//
// May be set separately in each "obs.field" entry
//
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
```

(continues on next page)

(continued from previous page)

```
//
mask = {
   ${METPLUS_MASK_GRID}
   ${METPLUS_MASK_POLY}
   sid   = [];
   llpnt = [];
}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
${METPLUS_CI_ALPHA}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gridded verification output types
// May be set separately in each "obs.field" entry
//
${METPLUS_NC_ORANK_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Random number generator
//
rng = {
   type = "mt19937";
   seed = "1";
```

(continues on next page)

(continued from previous page)

```
}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

${METPLUS_OUTPUT_PREFIX}
//version          = "V9.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

GridStatConfig_wrapped

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
```

(continues on next page)

(continued from previous page)

```
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
```

(continues on next page)

(continued from previous page)

```
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
```

(continues on next page)

(continued from previous page)

```
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in EnsembleStat_fcstWOFS_obsWOFS.py then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪EnsembleStat_fcstWOFS_obsWOFS.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in EnsembleStat_fcstWOFS_obsWOFS.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪EnsembleStat_fcstWOFS_obsWOFS.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in WOFS/grid_stat (relative to **OUTPUT_BASE**) The following folder/file combination will be created:

- 20200615/1700/grid_stat_WOFS_PCP_1700_A1_000000L_20200615_180000V_pairs.nc
- 20200615/1700/grid_stat_WOFS_PCP_1700_A1_000000L_20200615_180000V.stat

Keywords

Note:

- EnsembleStatToolUseCase
- PrecipitationAppUseCase
- GRIB2FileUseCase
- EnsembleAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/precipitation-EnsembleStat_fcstWOFS_obsWOFS.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.9.4 Grid-Stat: 6hr PQPF Probability Verification

model_applications/precipitation/GridStat_fcstHRRR-TLE _obsStgIV_GRIB.conf

Scientific Objective

This use case demonstrates the evaluation of a probabilistic field. The HRRR-Time Lag Ensemble (TLE) used in this example was used to demonstrate prototype ensemble post-processing techniques. A time-lagged ensemble can provide higher temporal resolution and be used to compute several different accumulation amounts based on what data is available for each run time. 6 hour and 1 hour observation data is available at 6Z, so the 6 hour accumulation data is used. However, at 7Z only a 1 hour accumulation field is available, so it uses the 1 hour field, then steps back in time trying to build a 6 hour accumulation with earlier data. METplus is configured to only allow 1 hour or 6 hour accumulations in the input files, so a set of six 1 hour accumulation fields are combined to create a 6 hour accumulation field. The result is compared to the 6 hour forecast data.

Datasets

Relevant information about the datasets that would be beneficial include:

- Forecast dataset: HRRR-TLE probabilistic forecasts in GRIB2
- Observation dataset: Stage IV GRIB 1 and 6 hour precipitation accumulation

METplus Components

This use case first runs PCPCombine on the observation data to build a 6 hour precipitation accumulation from 1 hour files or a single 6 hour file. Then the observation data is regridded to the model grid using the RegridDataPlane. Finally, the observation files are compared to the forecast data using GridStat.

METplus Workflow

The following tools are used for each run time:

PCPCombine (observation) > RegridDataPlane (observation) > GridStat

This example loops by initialization time. For each initialization time it will process forecast leads 6 and 7. There is only one initialization time in this example, so the following will be run:

Run times:

Init: 2016-09-04_12Z
Forecast lead: 6

Init: 2016-09-04_12Z
Forecast lead: 7

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/precipitation/GridStat_fcstHRRR-TLE_obsStgIV_GRIB.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/precipitation/
↪GridStat_fcstHRRR-TLE_obsStgIV_GRIB.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###
```

(continues on next page)

(continued from previous page)

```
PROCESS_LIST = PCPCombine, RegridDataPlane, GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT

INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2016090412
INIT_END=2016090412
INIT_INCREMENT=60

LEAD_SEQ = 6, 7


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

# PCPCombine

OBS_PCP_COMBINE_RUN = True

OBS_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/StageIV
OBS_PCP_COMBINE_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/ST4.{valid?fmt=%Y%m%d%H}.{level?fmt=%HH}h

OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_
↪fcstHRRR-TLE_obsStgIV_GRIB/StageIV_grib/bucket
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/ST4.{valid?fmt=%Y%m%d%H}_A{level?fmt=
↪%HH}h


#RegridDataPlane
```

(continues on next page)

(continued from previous page)

```
OBS_REGRID_DATA_PLANE_RUN = True

OBS_REGRID_DATA_PLANE_INPUT_DIR = {OBS_PCP_COMBINE_OUTPUT_DIR}
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_
↪fcstHRRR-TLE_obsStgIV_GRIB/StageIV_grib/regrid
OBS_REGRID_DATA_PLANE_TEMPLATE =  {OBS_PCP_COMBINE_OUTPUT_TEMPLATE}


# GridStat

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/PHPT
FCST_GRID_STAT_INPUT_TEMPLATE= {init?fmt=%Y%m%d}/{init?fmt=%Y%m%d}_i{init?fmt=%H}_f{lead?fmt=
↪%HHH}_HRRRTLE_PHPT.grb2

OBS_GRID_STAT_INPUT_DIR = {OBS_REGRID_DATA_PLANE_OUTPUT_DIR}
OBS_GRID_STAT_INPUT_TEMPLATE =    {OBS_REGRID_DATA_PLANE_TEMPLATE}

GRID_STAT_VERIFICATION_MASK_TEMPLATE = {INPUT_BASE}/model_applications/precipitation/mask/
↪CONUS_HRRRTLE.nc, {INPUT_BASE}/model_applications/precipitation/mask/EAST_HRRRTLE.nc,
↪{INPUT_BASE}/model_applications/precipitation/mask/WEST_HRRRTLE.nc

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_fcstHRRR-TLE_
↪obsStgIV_GRIB/GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H%M}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = PHPT
OBTYPE = STAGE4_GRIB

FCST_IS_PROB = true
FCST_PROB_IN_GRIB_PDS = True

BOTH_VAR1_NAME = APCP
BOTH_VAR1_LEVELS = A06
BOTH_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2, gt152.4


###
# PCPCombine Settings
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

OBS_PCP_COMBINE_METHOD = ADD

OBS_PCP_COMBINE_INPUT_DATATYPE = GRIB
OBS_PCP_COMBINE_INPUT_ACCUMS = 6, 1


###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

REGRID_DATA_PLANE_METHOD = BUDGET
REGRID_DATA_PLANE_WIDTH = 2

REGRID_DATA_PLANE_VERIF_GRID = {INPUT_BASE}/model_applications/precipitation/mask/CONUS_
↪HRRRTLE.nc


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_OUTPUT_PREFIX = PROB_{MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}_
↪{CURRENT_FCST_LEVEL}

GRID_STAT_MASK_GRID =

GRID_STAT_OUTPUT_FLAG_PCT = BOTH
GRID_STAT_OUTPUT_FLAG_PSTD = BOTH
GRID_STAT_OUTPUT_FLAG_PJC = BOTH
GRID_STAT_OUTPUT_FLAG_PRC = BOTH
GRID_STAT_OUTPUT_FLAG_ECLV = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
```

(continues on next page)

(continued from previous page)

```
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
```

(continues on next page)

(continued from previous page)

```
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstHRRR-TLE_obsStgIV_GRIB.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪GridStat_fcstHRRR-TLE_obsStgIV_GRIB.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstHRRR-TLE_obsStgIV_GRIB.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪GridStat_fcstHRRR-TLE_obsStgIV_GRIB.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/precipitation/GridStat_fcstHRRR-TLE_obsStgIV_GRIB/grid_stat/201609041200 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_PROB_PHPT_APCP_vs_STAGE4_GRIB_APCP_A06_060000L_20160904_180000V_pct.txt
- grid_stat_PROB_PHPT_APCP_vs_STAGE4_GRIB_APCP_A06_060000L_20160904_180000V_pjc.txt
- grid_stat_PROB_PHPT_APCP_vs_STAGE4_GRIB_APCP_A06_060000L_20160904_180000V_prc.txt
- grid_stat_PROB_PHPT_APCP_vs_STAGE4_GRIB_APCP_A06_060000L_20160904_180000V_pstd.txt
- grid_stat_PROB_PHPT_APCP_vs_STAGE4_GRIB_APCP_A06_060000L_20160904_180000V.stat
- grid_stat_PROB_PHPT_APCP_vs_STAGE4_GRIB_APCP_A06_060000L_20160904_190000V_pct.txt
- grid_stat_PROB_PHPT_APCP_vs_STAGE4_GRIB_APCP_A06_060000L_20160904_190000V_pjc.txt
- grid_stat_PROB_PHPT_APCP_vs_STAGE4_GRIB_APCP_A06_060000L_20160904_190000V_prc.txt
- grid_stat_PROB_PHPT_APCP_vs_STAGE4_GRIB_APCP_A06_060000L_20160904_190000V_pstd.txt
- grid_stat_PROB_PHPT_APCP_vs_STAGE4_GRIB_APCP_A06_070000L_20160904_190000V.stat

Keywords

Note:

- GridStatToolUseCase
- PrecipitationAppUseCase
- PCPCombineToolUseCase
- RegridDataPlaneToolUseCase
- GRIBFileUseCase
- GRIB2FileUseCase
- NetCDFFileUseCase
- NOAAWPCOrgUseCase
- NOAAHMTOrgUseCase
- NOAAHWTOrgUseCase
- ConvectionAllowingModelsAppUseCase
- ProbabilityVerificationUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/precipitation-GridStat_fcstHRRR-TLE_obsStgIV_GRIB.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.9.5 Point-Stat: Investigating Preciptitation Types

model_application/precipitation/PointStat_fcstMULTI_obsMETAR_PtypeComparisons.conf

Scientific Objective

During a storm that produces mulitiple precipitation types, the validation process becomes critical to investigating how well a model does during these situations. Using METplus' PointStat tool in this use case creates an opportunity to compare three separate model outputs for a multi-precipitation type storm across several valid times and create statistical output that can help modelers fine-tune curent numerical models to perform better in this forecast situation.

Datasets

- Forecast dataset: operational GFS, GFSv16, NAM
- Observation dataset: METARs (via NAM prepbufr reanalysis)

METplus Components

This use case runs PB2NC on each NAM prepbufr file, extracts the METAR data within a 30-minute window of the valid time, then runs Point-Stat on the model forecasts, comparing each valid time to the newly created netCDFs.

METplus Workflow

The following tools are used for each run time: PB2NC, PointStat

This example loops by initialization time. For each initialization time it will process the listed lead hours (12 hour steps from 12 to 84 hours)

Run times:

Init: 2021-02-15_12Z
Forecast leads: 12, 24, 36, 48, 60, 72, 84 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/precipitation/PointStat_fcstMULTI_obsMETAR_PtypeComparisons.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/precipitation/
↪PointStat_fcstMULTI_obsMETAR_PtypeComparisons.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###
```

(continues on next page)

(continued from previous page)

```
#The first PointStat call is for the GFS
PROCESS_LIST = PB2NC,PointStat,PointStat(nam_run),PointStat(gfsx_run)


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H%M
INIT_BEG = 202102151200
INIT_END = 202102151200
INIT_INCREMENT = 12H

LEAD_SEQ = 12, 24, 36, 48, 60, 72, 84


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

# PB2NC
PB2NC_OFFSETS = 0, 12

PB2NC_SKIP_IF_OUTPUT_EXISTS = True

PB2NC_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/PointStat_fcstMULTI_obsMETAR_
↪PtypeComparisons
PB2NC_INPUT_TEMPLATE = nam.{valid?fmt=%Y%m%d}.t{valid?fmt=%H}z.prepbufr.tm00

PB2NC_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation
PB2NC_OUTPUT_TEMPLATE = nam.obsfile_sfc_prwe.{valid?fmt=%m%d%Y}_{valid?fmt=%H}z.nc
```

(continues on next page)

(continued from previous page)

```
# PointStat

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/PointStat_
↪fcstMULTI_obsMETAR_PtypeComparisons
FCST_POINT_STAT_INPUT_TEMPLATE = gfs.t12z.pgrb2.0p25.f{lead?fmt=%3H}

OBS_POINT_STAT_INPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation
OBS_POINT_STAT_INPUT_TEMPLATE = {PB2NC_OUTPUT_TEMPLATE}

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

FCST_VAR1_NAME = CRAIN
FCST_VAR1_LEVELS = L0
FCST_VAR1_THRESH = >=1.0

OBS_VAR1_NAME = PRWE
OBS_VAR1_LEVELS = Z0
OBS_VAR1_THRESH = >=161&&<=163

FCST_VAR2_NAME = CSNOW
FCST_VAR2_LEVELS = L0
FCST_VAR2_THRESH = >=1.0

OBS_VAR2_NAME = PRWE
OBS_VAR2_LEVELS = Z0
OBS_VAR2_THRESH = >=171&&<=173

FCST_VAR3_NAME = CFRZR
FCST_VAR3_LEVELS = L0
FCST_VAR3_THRESH = >=1.0

OBS_VAR3_NAME = PRWE
OBS_VAR3_LEVELS = Z0
OBS_VAR3_THRESH = >=164&&<=166

FCST_VAR4_NAME = CICEP
FCST_VAR4_LEVELS = L0
FCST_VAR4_THRESH = >=1.0
```

(continues on next page)

(continued from previous page)

```
OBS_VAR4_NAME = PRWE
OBS_VAR4_LEVELS = Z0
OBS_VAR4_THRESH = >=174&&<=176


###
# PB2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pb2nc
###

PB2NC_OBS_WINDOW_BEGIN = -1800
PB2NC_OBS_WINDOW_END = 1800

PB2NC_GRID =
PB2NC_POLY =
PB2NC_STATION_ID =
PB2NC_MESSAGE_TYPE = ADPSFC

PB2NC_PB_REPORT_TYPE =

PB2NC_LEVEL_CATEGORY =

PB2NC_QUALITY_MARK_THRESH = 2

# Leave empty to process all
PB2NC_OBS_BUFR_VAR_LIST = PRWE

PB2NC_TIME_SUMMARY_FLAG = False
PB2NC_TIME_SUMMARY_BEG = 000000
PB2NC_TIME_SUMMARY_END = 235959
PB2NC_TIME_SUMMARY_VAR_NAMES =
PB2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80

PB2NC_TIME_SUMMARY_RAW_DATA = False
PB2NC_TIME_SUMMARY_STEP = 3600
PB2NC_TIME_SUMMARY_WIDTH = 3600
PB2NC_TIME_SUMMARY_GRIB_CODES =
PB2NC_TIME_SUMMARY_VALID_FREQ = 0
PB2NC_TIME_SUMMARY_VALID_THRESH = 0.0


###
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###
```

(continues on next page)

(continued from previous page)

```
MODEL = gfs

OBS_POINT_STAT_WINDOW_BEGIN = -1800
OBS_POINT_STAT_WINDOW_END = 1800

POINT_STAT_OUTPUT_PREFIX = gfs

POINT_STAT_MESSAGE_TYPE = ADPSFC

POINT_STAT_MASK_POLY = MET_BASE/poly/CONUS.poly

POINT_STAT_OUTPUT_FLAG_CTC = STAT
POINT_STAT_OUTPUT_FLAG_CTS = STAT

[gfsx_run]
MODEL = gfsx
FCST_POINT_STAT_INPUT_TEMPLATE= gfsx.t12z.pgrb2.0p25.f{lead?fmt=%3H}
POINT_STAT_OUTPUT_PREFIX = gfsx

[nam_run]
MODEL = nam
FCST_POINT_STAT_INPUT_TEMPLATE=nam.t12z.awip32{lead?fmt=%2H}.tm00.grib2
POINT_STAT_OUTPUT_PREFIX = nam
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

PB2NCConfig_wrapped

Note: See the *PB2NC MET Configuration* (page 198) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// PB2NC configuration file.
```

(continues on next page)

(continued from previous page)

```
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// PrepBufr message type
//
${METPLUS_MESSAGE_TYPE}

//
// Mapping of message type group name to comma-separated list of values
// Derive PRMSL only for SURFACE message types
//
message_type_group_map = [
   { key = "SURFACE"; val = "ADPSFC,SFCSHP,MSONET";               },
   { key = "ANYAIR";  val = "AIRCAR,AIRCFT";                      },
   { key = "ANYSFC";  val = "ADPSFC,SFCSHP,ADPUPA,PROFLR,MSONET"; },
   { key = "ONLYSF";  val = "ADPSFC,SFCSHP";                      }
];

//
// Mapping of input PrepBufr message types to output message types
//
message_type_map = [];

//
// PrepBufr station ID
//
${METPLUS_STATION_ID}


////////////////////////////////////////////////////////////////////////////////

//
// Observation time window
//
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Observation retention regions
//
${METPLUS_MASK_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Observing location elevation
//
elevation_range = {
   beg =  -1000;
   end = 100000;
}

////////////////////////////////////////////////////////////////////////////////

//
// Observation types
//
//pb_report_type =
${METPLUS_PB_REPORT_TYPE}

in_report_type  = [];

instrument_type = [];

////////////////////////////////////////////////////////////////////////////////

//
// Vertical levels to retain
//
//level_range = {
${METPLUS_LEVEL_RANGE_DICT}

//level_category =
${METPLUS_LEVEL_CATEGORY}

////////////////////////////////////////////////////////////////////////////////

//
// BUFR variable names to retain or derive.
// If emtpy, process all available variables.
//
${METPLUS_OBS_BUFR_VAR}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Mapping of BUFR variable name to GRIB name. The default map is defined at
// obs_prepbufr_map. This replaces/expends the default map.
//
//obs_bufr_map =
${METPLUS_OBS_BUFR_MAP}

// This map is for PREPBUFR. It will be added into obs_bufr_map.
// Please do not override this map.
//obs_prepbufr_map =


////////////////////////////////////////////////////////////////////////////////

//quality_mark_thresh =
${METPLUS_QUALITY_MARK_THRESH}

event_stack_flag    = TOP;

////////////////////////////////////////////////////////////////////////////////
//
// Time periods for the summarization
//
${METPLUS_TIME_SUMMARY_DICT}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

//version = "V9.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

PointStatConfig_wrapped

Note: See the *PointStat MET Configuration* (page 218) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
```

(continues on next page)

(continued from previous page)

```
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
```

(continues on next page)

(continued from previous page)

```
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}
```

(continues on next page)

(continued from previous page)

```
//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in PointStat_fcstMULTI_obsMETAR_PtypeComparisons.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪PointStat_fcstMULTI_obsMETAR_PtypeComparisons.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in Point-

Stat_fcstMULTI_obsMETAR_PtypeComparisons.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪PointStat_fcstMULTI_obsMETAR_PtypeComparisons.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/precipitation (relative to **OUTPUT_BASE**) The following PB2NC output files will be created:

- nam.obsfile_sfc_prwe.02[dd]2021_[hh].nc

Where [dd] and [hh] corespond to each valid time run (total of 7 files).

The following PointStat output files will also be created in model_applications/precipitation (relative to **OUTPUT_BASE**):

- point_stat_[model]_[lead]0000L_[valid_YYMMDD_time]_[valid_HH_time].stat

Where [model] is gfs, gfsx (for gfsv16), or nam, and valid times correspond to the 7 valid times being processed (total of 21 files).

Keywords

Note:

- PointStatToolUseCase
- PB2NCToolUseCase
- PrecipitationAppUseCase
- GRIB2FileUseCase
- prepBUFRFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/precipitation-PointStat_fcstMULTI_obsMETAR_PtypeComparisons.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.9.6 Grid-Stat: 6hr QPF in GEMPAK format

model_applications/precipitation/GridStat_fcstHREFmean _obsStgIV_Gempak.conf

Scientific Objective

Evaluate the skill of a high resolution multi-model ensemble mean at predicting 6 hour precipitation accumulation using the NCEP Stage IV gauge corrected analysis.

Datasets

Relevant information about the datasets that would be beneficial include:

- Forecast dataset: HREF mean forecasts in Gempak
- Observation dataset: Stage IV GRIB 6 hour precipitation accumulation
- Sources of data (links, contacts, etc. . .)

External Dependencies

GempakToCF.jar

GempakToCF is an external tool that utilizes the Unidata NetCDF-Java package. The jar file that can be used to run the utility is available here: https://dtcenter.org/sites/default/files/community-code/metplus/utilities/GempakToCF.jar

To enable Gempak support, you must set [exe] GEMPAKTOCF_JAR in your user METplus configuration file:

[exe] GEMPAKTOCF_JAR = /path/to/GempakToCF.jar

See the GempakToCF use case for more information:

parm/use_cases/met_tool_wrapper/GempakToCF/GempakToCF.conf

More information on the package used to create the file is here: https://www.unidata.ucar.edu/software/netcdf-java

METplus Components

This use case first runs PCPCombine on the forecast data to build a 6 hour precipitation accumulation from 1 hour files or a single 6 hour file. Then the observation data is regridded to the model grid using the RegridDataPlane. Finally, the observation files are compared to the forecast data using GridStat.

METplus Workflow

The following tools are used for each run time:

PCPCombine (observation) > RegridDataPlane (observation) > GridStat

This example loops by initialization time. There is only one initalization time in this example so the following will be run:

Run times:

Init: 2017-05-09_12Z
Forecast lead: 18

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/precipitation/GridStat_fcstHREFmean_obsStgIV_Gempak.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/precipitation/
↪GridStat_fcstHREFmean_obsStgIV_Gempak.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PCPCombine, RegridDataPlane, GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2017050912
INIT_END=2017050912
INIT_INCREMENT=43200

LEAD_SEQ = 18


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###


FCST_PCP_COMBINE_RUN = True

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/HREFv2_Mean_Gempak
FCST_PCP_COMBINE_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/hrefmean_{init?fmt=%Y%m%d%H}f{lead?fmt=
↪%HHH}.grd

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_
↪fcstHREFmean_obsStgIV_Gempak/HREFv2_Mean/bucket
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/hrefmean_{valid?fmt=%Y%m%d%H}_A{level?
↪fmt=%HH}.nc


OBS_REGRID_DATA_PLANE_RUN = True

OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/StageIV
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = {valid?fmt=%Y%m%d?shift=-12H}12_st4.nc

OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_
↪fcstHREFmean_obsStgIV_Gempak/StageIV_gempak/regrid
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}_st4_A06.nc


FCST_GRID_STAT_INPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
FCST_GRID_STAT_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/hrefmean_{valid?fmt=%Y%m%d%H}_A{level?fmt=
↪%HH}.nc

OBS_GRID_STAT_INPUT_DIR = {OBS_REGRID_DATA_PLANE_OUTPUT_DIR}
OBS_GRID_STAT_INPUT_TEMPLATE = {OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE}

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_fcstHREFmean_
↪obsStgIV_Gempak/GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H%M}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = HREF_MEAN
OBTYPE = STAGE4
```

(continues on next page)

(continued from previous page)

```
FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A06
FCST_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2, gt152.4

OBS_VAR1_NAME = P06M_NONE
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2, gt152.4

OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = "({valid?fmt=%Y%m%d_%H%M%S},*,*)"

OBS_PCP_COMBINE_INPUT_DATATYPE = NETCDF


###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

FCST_PCP_COMBINE_METHOD = ADD

FCST_PCP_COMBINE_INPUT_DATATYPE = GEMPAK

FCST_PCP_COMBINE_CONSTANT_INIT = true

FCST_PCP_COMBINE_INPUT_ACCUMS = 1
FCST_PCP_COMBINE_INPUT_NAMES = P01M_NONE
FCST_PCP_COMBINE_INPUT_LEVELS = "(0,*,*)"


###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

REGRID_DATA_PLANE_METHOD = BUDGET
REGRID_DATA_PLANE_WIDTH = 2

REGRID_DATA_PLANE_VERIF_GRID={INPUT_BASE}/model_applications/precipitation/mask/CONUS_
↪HRRRTLE.nc


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
```

(continues on next page)

(continued from previous page)

```
###

GRID_STAT_REGRID_TO_GRID = OBS

GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 7, 15
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_DMAP = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE
GRID_STAT_NC_PAIRS_FLAG_DISTANCE_MAP = TRUE
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
```

(continues on next page)

(continued from previous page)

```
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
```

(continues on next page)

(continued from previous page)

```
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
```

(continues on next page)

(continued from previous page)

```
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
```

(continues on next page)

(continued from previous page)

```
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstHREFmean_obsStgIV_Gempak.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪GridStat_fcstHREFmean_obsStgIV_Gempak.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstHREFmean_obsStgIV_Gempak.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪GridStat_fcstHREFmean_obsStgIV_Gempak.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/precipitation/GridStat_fcstHREFmean_obsStgIV_Gempak/GridStat/201705091200 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_000000L_20170510_060000V_eclv.txt
- grid_stat_000000L_20170510_060000V_grad.txt
- grid_stat_000000L_20170510_060000V.stat

Keywords

Note:

- GridStatToolUseCase
- PrecipitationAppUseCase
- PCPCombineToolUseCase
- RegridDataPlaneToolUseCase
- GEMPAKFileUseCase
- NetCDFFileUseCase
- NOAAWPCOrgUseCase
- NOAAHMTOrgUseCase
- NOAAHWTOrgUseCase
- ConvectionAllowingModelsAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/precipitation-GridStat_fcstHREFmean_obsStgIV_Gempak.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.9.7 Grid-Stat: 24-hour QPF Use Case

model_applications/precipitation/GridStat_fcstGFS _obsCCPA_Grib.conf

Scientific Objective

To evaluate 24 hour precipitation over the United States using the NCEP Climatology Calibrated Precipitation Analysis (CCPA) generated by a global weather model.

Datasets

Relevant information about the datasets that would be beneficial include:

- Forecast dataset: GFS
- Observation dataset: Climatologically Calibrated Precipitation Analysis (CCPA)
- Sources of data (links, contacts, etc. . .)

METplus Components

This use case first runs PCPCombine on the observation data to build a 24 hour precipitation accumulation file. Then the observation data are compared to the forecast data using GridStat.

METplus Workflow

The following tools are used for each run time:

PCPCombine (observation) > GridStat

This example loops by valid time. There is only one initialization time in this example, so the following will be run:

Run times:

Init: 2017-06-13_00Z
Forecast lead: 24

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/precipitation/GridStat_fcstGFS_obsCCPA_GRIB.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/precipitation/
↪GridStat_fcstGFS_obsCCPA_GRIB.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PCPCombine, GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2017061300
VALID_END = 2017061300
VALID_INCREMENT = 86400

LEAD_SEQ = 24


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

FCST_PCP_COMBINE_RUN = True

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/fcst
FCST_PCP_COMBINE_INPUT_TEMPLATE = pgbf{lead?fmt=%HHH}.gfs.{init?fmt=%Y%m%d%H}

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_
↪fcstGFS_obsCCPA_GRIB/gfs/bucket
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = gfs.{init?fmt=%Y%m%d%H}_A{level?fmt=%HH}h

FCST_GRID_STAT_INPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
FCST_GRID_STAT_INPUT_TEMPLATE = gfs.{init?fmt=%Y%m%d%H}_A{level?fmt=%HH}h

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/daily_1deg_ccpa
OBS_GRID_STAT_INPUT_TEMPLATE = ccpa_conus_1.0d_{valid?fmt=%Y%m%d}

GRID_STAT_MASK_POLY = {INPUT_BASE}/model_applications/precipitation/poly/CONUS.nc, {INPUT_
↪BASE}/model_applications/precipitation/poly/EAST.nc, {INPUT_BASE}/model_applications/
↪precipitation/poly/WEST.nc

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_fcstGFS_
↪obsCCPA_GRIB/met_out/{MODEL}/precip
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H%M}/grid_stat


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = GFS
OBTYPE = ANLYS

BOTH_VAR1_NAME = APCP
BOTH_VAR1_LEVELS = A24
BOTH_VAR1_THRESH = ge12.7, ge25.4, ge50.8, ge76.2, ge152.4


###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###
```

(continues on next page)

(continued from previous page)

```
FCST_PCP_COMBINE_METHOD = SUM
FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB
FCST_PCP_COMBINE_INPUT_ACCUMS = 6


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_REGRID_TO_GRID =  G211
GRID_STAT_REGRID_METHOD = BILIN
GRID_STAT_REGRID_WIDTH = 2

GRID_STAT_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}_
↪{CURRENT_FCST_LEVEL}

GRID_STAT_CLIMO_CDF_WRITE_BINS = False

GRID_STAT_OUTPUT_FLAG_CTC = STAT

GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE

GRID_STAT_CLIMO_MEAN_REGRID_METHOD = BILIN
GRID_STAT_CLIMO_MEAN_REGRID_WIDTH = 2
GRID_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
```

(continues on next page)

(continued from previous page)

```
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
```

(continues on next page)

(continued from previous page)

```
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstGFS_obsCCPA_GRIB.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪GridStat_fcstGFS_obsCCPA_GRIB.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstGFS_obsCCPA_GRIB:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪GridStat_fcstGFS_obsCCPA_GRIB.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/precipitation/GridStat_fcstGFS_obsCCPA_GRIB/uswrp/met_out/{MODEL}/precip (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_GFS_APCP_vs_ANLYS_APCP_A24_240000L_20170613_000000V.stat

Keywords

Note:

- GridStatToolUseCase
- PrecipitationAppUseCase
- PCPCombineToolUseCase
- GRIBFileUseCase
- NOAAEMCOrgUseCase
- MediumRangeAppUseCase
- MaskingFeatureUseCase
- RegriddingInToolUseCase
- ClimatologyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/precipitation-GridStat_fcstGFS_obsCCPA_GRIB.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.9.8 MTD: Build Revision Series to Evaluate Forecast Consistency

model_applications/precipitation/MTD_fcstHRRR-TLE_FcstOnly _RevisionSeries_GRIB.conf

Scientific Objective

This use case demonstrates the use of the MTD tool to evaluate an updating forecast field and evaluate the forecast consistency. The use case looks for all forecasts valid at a given time and passes them into MTD. Objects are identified and tracked through time via the tool. The output can then be loaded into METviewer to compute the revision series and assess the consistency either of one case or many. See other HRRR-TLE use cases for a description of the Time Lagged Ensemble (TLE) field.

Datasets

- Forecast dataset: HRRR-TLE forecasts in GRIB2

METplus Components

This use case runs MTD (MODE Time Domain) over multiple forecast leads.

METplus Workflow

The following tools are used for each run time:

MTD

This example loops by valid time. For each valid time it will run once, processing forecast leads 12 through 0. There is only one valid time in this example, so the following will be run:

Run times:

Valid: 2018-03-13_0Z
Forecast leads: 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/precipitation/MTD_fcstHRRR-TLE_FcstOnly_RevisionSeries_GRIB.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/precipitation/MTD_
↪fcstHRRR-TLE_FcstOnly_RevisionSeries_GRIB.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = MTD


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG=2018031300
VALID_END=2018031300
VALID_INCREMENT=86400

LEAD_SEQ = begin_end_incr(12, 0, -1)


###
# File I/O
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_MTD_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/PHPT
FCST_MTD_INPUT_TEMPLATE= {init?fmt=%Y%m%d}/{init?fmt=%Y%m%d}_i{init?fmt=%H}_f{lead?fmt=%3H}_
↪HRRRTLE_PHPT.grb2

MTD_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/MTD_fcstHRRR-TLE_FcstOnly_
↪RevisionSeries_GRIB
MTD_OUTPUT_TEMPLATE =


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = HRRRTLE
OBTYPE = ANALYS

FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = R001

FCST_MTD_CONV_RADIUS = 15
FCST_MTD_CONV_THRESH = >=5.0
FCST_MTD_INPUT_DATATYPE = GRIB


###
# MTD Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mtd
###

MTD_SINGLE_RUN = True
MTD_SINGLE_DATA_SRC = FCST

MTD_REGRID_TO_GRID = NONE

MTD_MIN_VOLUME = 2000
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *MTD MET Configuration* (page 188) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE Time Domain configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

   //
   // Output model name to be written
   //

${METPLUS_MODEL}

   //
   // Output description to be written
   //

${METPLUS_DESC}

   //
   // Output observation type to be written
   //

${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

   //
   //  Verification grid
   //  May be set separately in each "field" entry
   //
```

(continues on next page)

(continued from previous page)

```
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

   //
   // Approximate grid resolution (km)
   //

grid_res = 4;

////////////////////////////////////////////////////////////////////////////////

   //
   // Forecast and observation fields to be verified
   //

fcst = {

   ${METPLUS_FCST_FILE_TYPE}

   ${METPLUS_FCST_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}

}

obs = {

   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
   //
   // Intensity percentile value to be written
   //

inten_perc_value = 99;

////////////////////////////////////////////////////////////////////////////////

   //
   //  Throw away 3D objects with volumes smaller than this
   //
${METPLUS_MIN_VOLUME}


////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine weights
   //

weight = {

   space_centroid_dist  = 1.0;

   time_centroid_delta  = 1.0;

   speed_delta          = 1.0;

   direction_diff       = 1.0;

   volume_ratio         = 1.0;

   axis_angle_diff      = 1.0;

   start_time_delta     = 1.0;

   end_time_delta       = 1.0;

}

////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine interest functions
   //
```

(continues on next page)

(continued from previous page)

```
interest_function = {

   space_centroid_dist = (

      (   0.0, 1.0 )
      (  50.0, 0.5 )
      ( 100.0, 0.0 )

   );

   time_centroid_delta = (

      ( -3.0, 0.0 )
      ( -2.0, 0.5 )
      ( -1.0, 0.8 )
      (  0.0, 1.0 )
      (  1.0, 0.8 )
      (  2.0, 0.5 )
      (  3.0, 0.0 )

   );

   speed_delta = (

      ( -10.0, 0.0 )
      (  -5.0, 0.5 )
      (   0.0, 1.0 )
      (   5.0, 0.5 )
      (  10.0, 0.0 )

   );

   direction_diff = (

      (   0.0, 1.0 )
      (  90.0, 0.0 )
      ( 180.0, 0.0 )

   );

   volume_ratio = (

      (  0.0, 0.0 )
      (  0.5, 0.5 )
```

(continues on next page)

(continued from previous page)

```
      (  1.0, 1.0 )
      (  1.5, 0.5 )
      (  2.0, 0.0 )

   );

   axis_angle_diff = (

      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 0.0 )

   );

   start_time_delta = (

      ( -5.0, 0.0 )
      ( -3.0, 0.5 )
      (  0.0, 1.0 )
      (  3.0, 0.5 )
      (  5.0, 0.0 )

   );

   end_time_delta = (

      ( -5.0, 0.0 )
      ( -3.0, 0.5 )
      (  0.0, 1.0 )
      (  3.0, 0.5 )
      (  5.0, 0.0 )

   );

}   //  interest functions


////////////////////////////////////////////////////////////////////////////////

   //
   // Total interest threshold for determining matches
   //

total_interest_thresh = 0.7;
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

   //
   // Output flags
   //

nc_output = {

   latlon       = true;
   raw          = true;
   object_id    = true;
   cluster_id   = true;

}

txt_output = {

   attributes_2d   = true;
   attributes_3d   = true;

}


////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in MTD_fcstHRRR-TLE_FcstOnly_RevisionSeries_GRIB.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/MTD_
↪fcstHRRR-TLE_FcstOnly_RevisionSeries_GRIB.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in MTD_fcstHRRR-

TLE_FcstOnly_RevisionSeries_GRIB.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/MTD_
↪fcstHRRR-TLE_FcstOnly_RevisionSeries_GRIB.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/precipitation/MTD_fcstHRRR-TLE_FcstOnly_RevisionSeries_GRIB (relative to **OUTPUT_BASE**) and will contain the following files:

- mtd_20180313_000000V_2d.txt
- mtd_20180313_000000V_3d_single_simple.txt
- mtd_20180313_000000V_obj.nc

Keywords

Note:

- MTDToolUseCase
- PrecipitationAppUseCase
- NOAAHMTOrgUseCase
- GRIB2FileUseCase
- NOAAWPCOrgUseCase
- NOAAHMTOrgUseCase
- NOAAHWTOrgUseCase
- ConvectionAllowingModelsAppUseCase
- RevisionSeriesUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/precipitation-MTD_fcstHRRR-TLE_FcstOnly_RevisionSeries_GRIB.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.9.9 MTD: 6hr QPF Use Case

model_applications/precipitation/MTD_fcstHRRR-TLE _obsMRMS.conf

Scientific Objective

This use case demonstrates the evaluation of an ensemble mean field from a prototype ensemble post-processing technique for time-lagged ensembles (HRRR-TLE). MTD is used to provide useful object attributes and diagnostics on aggregated over a time series. This non-traditional approach provides alternative information and diagnostics to inform model development.

Datasets

- Forecast dataset: HRRR-TLE forecasts in GRIB2
- Observation dataset: Multi Radar Multi Sensor (MRMS)

METplus Components

This use case runs MTD (MODE Time Domain) over multiple forecast leads and compares them to the observational data set.

METplus Workflow

The following tools are used for each run time:

MTD

This example loops by valid time. For each valid time it will run once, processing forecast leads 1, 2, and 3. There is only one valid time in this example, so the following will be run:

Run times:

Valid: 2017-05-10_03Z
Forecast leads: 1, 2, 3

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/precipitation/MTD_fcstHRRR-TLE_obsMRMS.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/precipitation/MTD_
↪fcstHRRR-TLE_obsMRMS.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
```

(continues on next page)

(continued from previous page)

```
###

PROCESS_LIST = MTD


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2017051003
INIT_END=2017051003
INIT_INCREMENT=43200

LEAD_SEQ = 1,2,3


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_MTD_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/PHPT
FCST_MTD_INPUT_TEMPLATE= {init?fmt=%Y%m%d}/{init?fmt=%Y%m%d}_i{init?fmt=%H}_f{lead?fmt=%HHH}_
↪HRRRTLE_PHPT.grb2

OBS_MTD_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/QPE_Data
OBS_MTD_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/qpe_{valid?fmt=%Y%m%d%H}.nc

MTD_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/MTD_fcstHRRR-TLE_obsMRMS
MTD_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H%M}


###
# Field Info
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = PHPT
OBTYPE = QPE

FCST_IS_PROB = true
FCST_PROB_IN_GRIB_PDS = true

FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A01
FCST_VAR1_THRESH = gt12.7

OBS_VAR1_NAME = P01M_NONE
OBS_VAR1_LEVELS = "(0,*,*)"
OBS_VAR1_THRESH = gt12.7


###
# MTD Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mtd
###

FCST_MTD_CONV_RADIUS = 0
FCST_MTD_CONV_THRESH = >=10

OBS_MTD_CONV_RADIUS = 15
OBS_MTD_CONV_THRESH = >=12.7

MTD_REGRID_TO_GRID = OBS

MTD_OUTPUT_PREFIX = PROB_{MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}_A
↪{CURRENT_FCST_LEVEL}
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *MTD MET Configuration* (page 188) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE Time Domain configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

   //
   // Output model name to be written
   //

${METPLUS_MODEL}

   //
   // Output description to be written
   //

${METPLUS_DESC}

   //
   // Output observation type to be written
   //

${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

   //
   //  Verification grid
   //  May be set separately in each "field" entry
   //
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

   //
   // Approximate grid resolution (km)
   //

grid_res = 4;
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

   //
   // Forecast and observation fields to be verified
   //

fcst = {

   ${METPLUS_FCST_FILE_TYPE}

   ${METPLUS_FCST_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}

}

obs = {

   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}

   censor_thresh     = [];
   censor_val        = [];
   conv_time_window  = { beg = -1; end = 1; };
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
}

////////////////////////////////////////////////////////////////////////////////

   //
   // Intensity percentile value to be written
   //

inten_perc_value = 99;

////////////////////////////////////////////////////////////////////////////////

   //
   //  Throw away 3D objects with volumes smaller than this
```

(continues on next page)

(continued from previous page)

```
   //
${METPLUS_MIN_VOLUME}


////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine weights
   //

weight = {

   space_centroid_dist  = 1.0;

   time_centroid_delta  = 1.0;

   speed_delta          = 1.0;

   direction_diff       = 1.0;

   volume_ratio         = 1.0;

   axis_angle_diff      = 1.0;

   start_time_delta     = 1.0;

   end_time_delta       = 1.0;

}

////////////////////////////////////////////////////////////////////////////////

   //
   // Fuzzy engine interest functions
   //

interest_function = {

   space_centroid_dist = (

      (   0.0, 1.0 )
      (  50.0, 0.5 )
      ( 100.0, 0.0 )

   );
```

(continues on next page)

(continued from previous page)

```
time_centroid_delta = (

   ( -3.0, 0.0 )
   ( -2.0, 0.5 )
   ( -1.0, 0.8 )
   (  0.0, 1.0 )
   (  1.0, 0.8 )
   (  2.0, 0.5 )
   (  3.0, 0.0 )

);

speed_delta = (

   ( -10.0, 0.0 )
   (  -5.0, 0.5 )
   (   0.0, 1.0 )
   (   5.0, 0.5 )
   (  10.0, 0.0 )

);

direction_diff = (

   (   0.0, 1.0 )
   (  90.0, 0.0 )
   ( 180.0, 0.0 )

);

volume_ratio = (

   (  0.0, 0.0 )
   (  0.5, 0.5 )
   (  1.0, 1.0 )
   (  1.5, 0.5 )
   (  2.0, 0.0 )

);

axis_angle_diff = (

   (  0.0, 1.0 )
   ( 30.0, 1.0 )
```

(continues on next page)

(continued from previous page)

```
      ( 90.0, 0.0 )

   );

   start_time_delta = (

      ( -5.0, 0.0 )
      ( -3.0, 0.5 )
      (  0.0, 1.0 )
      (  3.0, 0.5 )
      (  5.0, 0.0 )

   );

   end_time_delta = (

      ( -5.0, 0.0 )
      ( -3.0, 0.5 )
      (  0.0, 1.0 )
      (  3.0, 0.5 )
      (  5.0, 0.0 )

   );

}   //  interest functions


////////////////////////////////////////////////////////////////////////////////

   //
   // Total interest threshold for determining matches
   //

total_interest_thresh = 0.7;


////////////////////////////////////////////////////////////////////////////////

   //
   // Output flags
   //

nc_output = {

   latlon       = true;
```

(continues on next page)

(continued from previous page)

```
   raw          = true;
   object_id    = true;
   cluster_id   = true;

}

txt_output = {

   attributes_2d   = true;
   attributes_3d   = true;

}


////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in MTD_fcstHRRR-TLE_obsMRMS.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/MTD_
↪fcstHRRR-TLE_obsMRMS.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in MTD_fcstHRRR-TLE_obsMRMS.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/MTD_
↪fcstHRRR-TLE_obsMRMS.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases

- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/precipitation/MTD_fcstHRRR-TLE_obsMRMS (relative to **OUTPUT_BASE**) and will contain the following files:

- mtd_20170510_040000V_2d.txt
- mtd_20170510_040000V_3d_single_simple.txt
- mtd_20170510_040000V_obj.nc

Keywords

Note:

- MTDToolUseCase
- PrecipitationAppUseCase
- GRIB2FileUseCase
- NetCDFFileUseCase
- NOAAWPCOrgUseCase
- NOAAHMTOrgUseCase
- NOAAHWTOrgUseCase
- ConvectionAllowingModelsAppUseCase
- ProbabilityVerificationUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/precipitation-MTD_fcstHRRR-TLE_obsMRMS.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.9.10 PointStat: Compare community observed precipitation to model forecasts

model_applications/precipitation/PointStat_fcstURMA_obsCOCORAHS_ASCIIprecip.conf

Scientific Objective

This use case ingests a CoCoRaHS csv file, a new dataset that utilizes community reporting of precipitation amounts. Numerous studies have shown that a community approach to weather observations not only covers areas that lack traditional verification datasets, but is also remarkably quality controlled. Utilizing Python embedding, this use case taps into a new vital observation dataset and compares it to a 24 hour precipitation accumulation forecast.

Datasets

Forecast: 24 URMA 1 hour precipitation accumulation files

Observations: CoCoRaHS, the Community Collaborative Rain, Hail, and Snow Network

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1520) section for more information.

Data Source: EMC

METplus Components

This use case calls a Python script in ASCII2NC for the observation dataset. PCPCombine is called for a user-defined summation of the forecast accumulation fields. Finally, PointStat processes the forecast and observation fields, and outputs the requested line types.

METplus Workflow

1 csv file of multiple valid observation times is passed to ASCII2NC via Python embedding, resulting in a netCDF output. 24 forecast files, each composed of 1 hour precipitation accumulation forecasts, is summarized via PCPCombine. The following boundary times are used for the forecast summation times:

Valid Beg: 2022-09-14 at 00z
Valid End: 2022-09-14 at 23z

The observation data point span the same times as the 24 hour forecast accumulation summation. Finally, PointStat is used to compare the two new fields (point data in netCDF and precipitation accumulation over 24 hours). Because the Valid Time used in configuration file is set to one time (2022-09-14 at 23z) and the precipitation accumulation valid time is set to this same time, the observation window spans across the entire 2022-09-14 24 hour timeframe.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. -c parm/use_cases/model_applications/precipitation/PointStat_fcstURMA_obsCOCORAHS_ASCIIprecip.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/met_tool_wrapper/PointStat/PointStat.
↪html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = ASCII2NC, PCPCombine, PointStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
```

(continues on next page)

(continued from previous page)

```
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2022091423
VALID_END = 2022091423
VALID_INCREMENT = 1M

LEAD_SEQ = 24H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###


ASCII2NC_INPUT_DIR =
ASCII2NC_INPUT_TEMPLATE = "{PARM_BASE}/use_cases/model_applications/precipitation/PointStat_
↪fcstURMA_obsCOCORAHS_ASCIIprecip/read_cocorahs_point.py {INPUT_BASE}/model_applications/
↪precipitation/PointStat_fcstURMA_obsCOCORAHS_ASCIIprecip/CoCoRaHS.{valid?fmt=%Y%m%d}.
↪DailyPrecip.csv"

ASCII2NC_OUTPUT_DIR =
ASCII2NC_OUTPUT_TEMPLATE = {OUTPUT_BASE}/ASCII2NC/precip_{valid?fmt=%Y%m%d}_summary.nc

ASCII2NC_SKIP_IF_OUTPUT_EXISTS = False

ASCII2NC_FILE_WINDOW_BEGIN = 0
ASCII2NC_FILE_WINDOW_END = 0


FCST_PCP_COMBINE_RUN = True

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/PointStat_
↪fcstURMA_obsCOCORAHS_ASCIIprecip
FCST_PCP_COMBINE_INPUT_TEMPLATE = {lead?fmt=%HH}

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/PCPCombine
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = fcst_24hr_precip.nc
```

(continues on next page)

(continued from previous page)

```
FCST_POINT_STAT_INPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
FCST_POINT_STAT_INPUT_TEMPLATE = {FCST_PCP_COMBINE_OUTPUT_TEMPLATE}

OBS_POINT_STAT_INPUT_DIR = {OUTPUT_BASE}/ASCII2NC
OBS_POINT_STAT_INPUT_TEMPLATE = precip_{valid?fmt=%Y%m%d}_summary.nc

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/PointStat

POINT_STAT_CLIMO_MEAN_INPUT_DIR =
POINT_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

POINT_STAT_CLIMO_STDEV_INPUT_DIR =
POINT_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

###
# ASCII2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ascii2nc
###

#LOG_ASCII2NC_VERBOSITY = 1
#ASCII2NC_CONFIG_FILE =

ASCII2NC_WINDOW_BEGIN = 0
ASCII2NC_WINDOW_END = 0

ASCII2NC_INPUT_FORMAT = python

ASCII2NC_MASK_GRID =
ASCII2NC_MASK_POLY =
ASCII2NC_MASK_SID =

ASCII2NC_TIME_SUMMARY_FLAG = False
ASCII2NC_TIME_SUMMARY_RAW_DATA = False
ASCII2NC_TIME_SUMMARY_BEG = 000000
ASCII2NC_TIME_SUMMARY_END = 235959
ASCII2NC_TIME_SUMMARY_STEP = 300
ASCII2NC_TIME_SUMMARY_WIDTH = 600
ASCII2NC_TIME_SUMMARY_GRIB_CODES =
ASCII2NC_TIME_SUMMARY_VAR_NAMES = APCP
ASCII2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80
ASCII2NC_TIME_SUMMARY_VALID_FREQ = 0
ASCII2NC_TIME_SUMMARY_VALID_THRESH = 0.0
```

(continues on next page)

(continued from previous page)

```
###
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

FCST_PCP_COMBINE_METHOD = USER_DEFINED

FCST_PCP_COMBINE_COMMAND = -sum 00000000_000000 1 20220914_230000 24 {FCST_PCP_COMBINE_
↪OUTPUT_DIR}/{FCST_PCP_COMBINE_OUTPUT_TEMPLATE} -pcpdir {FCST_PCP_COMBINE_INPUT_DIR}

#LOG_PCP_COMBINE_VERBOSITY = 2

FCST_IS_PROB = false

FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB

FCST_PCP_COMBINE_INPUT_ACCUMS = 1H
FCST_PCP_COMBINE_INPUT_NAMES = APCP
FCST_PCP_COMBINE_INPUT_LEVELS = L0

FCST_PCP_COMBINE_OUTPUT_ACCUM = 24

###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

POINT_STAT_ONCE_PER_FIELD = False

#POINT_STAT_FCST_FILE_TYPE =
#POINT_STAT_OBS_FILE_TYPE =

FCST_POINT_STAT_VAR1_NAME = APCP_24
FCST_POINT_STAT_VAR1_LEVELS = L0
#FCST_VAR1_THRESH = <=273, >273
OBS_POINT_STAT_VAR1_NAME = TotalPrecipAmt
OBS_POINT_STAT_VAR1_LEVELS = L0
#OBS_VAR1_THRESH = <=273, >273
BOTH_POINT_STAT_VAR1_THRESH = <=6.35, <=12.7, <=25.4

###
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###
```

(continues on next page)

(continued from previous page)

```
#LOG_POINT_STAT_VERBOSITY = 2

POINT_STAT_CONFIG_FILE ={PARM_BASE}/met_config/PointStatConfig_wrapped

#POINT_STAT_OBS_QUALITY_INC = 1, 2, 3
#POINT_STAT_OBS_QUALITY_EXC =

#POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST
#POINT_STAT_CLIMO_STDEV_TIME_INTERP_METHOD =

#POINT_STAT_INTERP_VLD_THRESH =
#POINT_STAT_INTERP_SHAPE =
#POINT_STAT_INTERP_TYPE_METHOD = BILIN
#POINT_STAT_INTERP_TYPE_WIDTH = 2

#POINT_STAT_OUTPUT_FLAG_FHO =
POINT_STAT_OUTPUT_FLAG_CTC = BOTH
POINT_STAT_OUTPUT_FLAG_CTS = BOTH
#POINT_STAT_OUTPUT_FLAG_MCTC =
POINT_STAT_OUTPUT_FLAG_MCTS = BOTH
POINT_STAT_OUTPUT_FLAG_CNT = BOTH
#POINT_STAT_OUTPUT_FLAG_SL1L2 = STAT
#POINT_STAT_OUTPUT_FLAG_SAL1L2 =
#POINT_STAT_OUTPUT_FLAG_VL1L2 = STAT
#POINT_STAT_OUTPUT_FLAG_VAL1L2 =
#POINT_STAT_OUTPUT_FLAG_VCNT =
#POINT_STAT_OUTPUT_FLAG_PCT =
#POINT_STAT_OUTPUT_FLAG_PSTD =
#POINT_STAT_OUTPUT_FLAG_PJC =
#POINT_STAT_OUTPUT_FLAG_PRC =
#POINT_STAT_OUTPUT_FLAG_ECNT =
#POINT_STAT_OUTPUT_FLAG_RPS =
#POINT_STAT_OUTPUT_FLAG_ECLV =
#POINT_STAT_OUTPUT_FLAG_MPR =
#POINT_STAT_OUTPUT_FLAG_ORANK =

#POINT_STAT_CLIMO_CDF_BINS = 1
#POINT_STAT_CLIMO_CDF_CENTER_BINS = False
#POINT_STAT_CLIMO_CDF_WRITE_BINS = True
#POINT_STAT_CLIMO_CDF_DIRECT_PROB =

#POINT_STAT_HSS_EC_VALUE =

OBS_POINT_STAT_WINDOW_BEGIN = -82800
OBS_POINT_STAT_WINDOW_END = 3600
```

(continues on next page)

(continued from previous page)

```
POINT_STAT_OFFSETS = 0

MODEL = URMA

POINT_STAT_DESC = CoCoRaHS
OBTYPE =

POINT_STAT_REGRID_TO_GRID = NONE
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2

POINT_STAT_OUTPUT_PREFIX =

#POINT_STAT_OBS_VALID_BEG = {valid?fmt=%Y%m%d_%H}
#POINT_STAT_OBS_VALID_END = {valid?fmt=%Y%m%d_%H}

POINT_STAT_MASK_GRID = FULL
POINT_STAT_MASK_POLY =
POINT_STAT_MASK_SID =
#POINT_STAT_MASK_LLPNT =

POINT_STAT_MESSAGE_TYPE = ADPSFC

#POINT_STAT_HIRA_FLAG =
#POINT_STAT_HIRA_WIDTH =
#POINT_STAT_HIRA_VLD_THRESH =
#POINT_STAT_HIRA_COV_THRESH =
#POINT_STAT_HIRA_SHAPE =
#POINT_STAT_HIRA_PROB_CAT_THRESH =

#POINT_STAT_MESSAGE_TYPE_GROUP_MAP =
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

```
////////////////////////////////////////////////////////////////////////////////
//
// Default ascii2nc configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to summarize the ASCII data read in
//

//
// Time periods for the summarization
// obs_var (string array) is added and works like grib_code (int array)
// when the obs name is given instead of grib_code
//
${METPLUS_TIME_SUMMARY_DICT}

//
// Mapping of input little_r report types to output message types
//
message_type_map = [
   { key = "FM-12 SYNOP";  val = "ADPSFC"; },
   { key = "FM-13 SHIP";   val = "SFCSHP"; },
   { key = "FM-15 METAR";  val = "ADPSFC"; },
   { key = "FM-18 BUOY";   val = "SFCSHP"; },
   { key = "FM-281 QSCAT"; val = "ASCATW"; },
   { key = "FM-32 PILOT";  val = "ADPUPA"; },
   { key = "FM-35 TEMP";   val = "ADPUPA"; },
   { key = "FM-88 SATOB";  val = "SATWND"; },
   { key = "FM-97 ACARS";  val = "AIRCFT"; }
];

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
```

(continues on next page)

(continued from previous page)

```
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

//
// Forecast and observation fields to be verified
//
```

(continues on next page)

(continued from previous page)

```
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Interpolation methods
//
//interp = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in PointStat_fcstURMA_obsCOCORAHS_ASCIIprecip.conf then a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪PointStat_fcstURMA_obsCOCORAHS_ASCIIprecip.conf /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in PointStat_fcstURMA_obsCOCORAHS_ASCIIprecip:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪PointStat_fcstURMA_obsCOCORAHS_ASCIIprecip.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[config]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for the use case will be found in 3 folders(relative to **OUTPUT_BASE**). Those folders are:

- ASCII2NC
- PCPCombine
- PointStat

The ASCII2NC folder will contain one file from the ASCII2NC tool call:

- precip_20220914_summary.nc

The PCPCombine folder will also contain one file, from the PCPCombine call:

- fcst_24hr_precip.nc

The final folder, PointStat, contains all of the following output from the PointStat call:

- point_stat_000000L_20220914_230000V_cnt.txt
- point_stat_000000L_20220914_230000V_ctc.txt

- point_stat_000000L_20220914_230000V_cts.txt
- point_stat_000000L_20220914_230000V_mcts.txt
- point_stat_000000L_20220914_230000V.stat

Keywords

Note:

- PointStatToolUseCase
- ASCII2NCToolUseCase
- PCPCombineToolUseCase
- PythonEmbeddingFileUseCase
- PrecipitationAppUseCase
- NETCDFFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/precipitation-PointStat_fcstURMA_obsCOCORAHS_ASCIIprecip.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.10 Subseasonal to Seasonal

Subseasonal-to-Seasonal model configurations; Lower resolution model configurations (>4km) usually producing forecasts out beyond 14 days and up 1 year

7.2.16.10.1 UserScript: Compue Cross Spectra and make a plot

model_applications/ s2s/ UserScript_fcstS2S_obsERAI_CrossSpectra.py

Scientific Objective

This use case calls the METcalcpy cross spectra function and then the METplotpy space time plot to compute cross-spectra and create a sample cross spectra diagram using sample data.

The space time plot and cross spectra calculations were created by Maria Gehne at the Physical Sciences Labratory in NOAA.

Datasets

- Forecast dataset: UFS Prototype 7
- Observation dataset: ERAI

METplus Components

This use case runs the UserScript wrapper tool to run a user provided script, in this case, cross_spectra.py and cross_spectra_plot.py.

METplus Workflow

This use case computes spectra and plots for the entire time period of data. The use case loops over two processes, computing and plotting the cross-spectra. The user is able to edit the process list to turn off the computation part if only plotting is desired, or vice versa.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/s2s/UserScript_fcstS2S_obsERAI_CrossSpectra.conf

```
[config]

PROCESS_LIST = UserScript(comp_spectra), UserScript(plot_spectra)

# Note: time looping is not used in this use case
LOOP_BY = REALTIME
VALID_TIME_FMT = %Y
VALID_BEG = 2014

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE

[user_env_vars]
# Make output base avabilable to the script
COMP_SPECTRA_SCRIPT_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstS2S_obsERAI_CrossSpectra/
↪output

# YAML configuration file for the cross spectra calculation
COMP_SPECTRA_YAML_CONFIG_NAME = {METPLUS_BASE}/parm/use_cases/model_applications/s2s/
↪UserScript_fcstS2S_obsERAI_CrossSpectra/spectra_comp.yaml

# Input files for the cross spectra calculation
```

(continues on next page)

(continued from previous page)

```
COMP_SPECTRA_INPUT_FILE_NAMES = {INPUT_BASE}/model_applications/s2s/UserScript_fcstS2S_
↪obsERAI_CrossSpectra/data/precip.erai.sfc.1p0.2x.2014-2016.nc,{INPUT_BASE}/model_
↪applications/s2s/UserScript_fcstS2S_obsERAI_CrossSpectra/data/prate_avg_ufs_p7_2014040100.
↪nc,{INPUT_BASE}/model_applications/s2s/UserScript_fcstS2S_obsERAI_CrossSpectra/data/u850_
↪ufs_p7_2014040100.nc,{INPUT_BASE}/model_applications/s2s/UserScript_fcstS2S_obsERAI_
↪CrossSpectra/data/u200_ufs_p7_2014040100.nc

PLOT_SPECTRA_INPUT_FILE_NAMES = {OUTPUT_BASE}/s2s/UserScript_fcstS2S_obsERAI_CrossSpectra/
↪output/SpaceTimeSpectra_ufs_p7_P_D850_symm_4spd.nc,{OUTPUT_BASE}/s2s/UserScript_fcstS2S_
↪obsERAI_CrossSpectra/output/SpaceTimeSpectra_ufs_p7_P_D200_symm_4spd.nc,{OUTPUT_BASE}/s2s/
↪UserScript_fcstS2S_obsERAI_CrossSpectra/output/SpaceTimeSpectra_ufs_p7_P_D200_symm_4spd.nc

PLOT_SPECTRA_YAML_CONFIG_NAME = {METPLUS_BASE}/parm/use_cases/model_applications/s2s/
↪UserScript_fcstS2S_obsERAI_CrossSpectra/spectra_plot.yaml

PLOT_SPECTRA_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstS2S_obsERAI_CrossSpectra/plots/


[comp_spectra]
# Settings for computing the cross-spectra
USER_SCRIPT_COMMAND = python {METCALCPY_BASE}/metcalcpy/contributed/spacetime/cross_spectra.
↪py

LOG_FILE = "cross_spectra.log"
LOG_LEVEL = "DEBUG"

METCALCPY_BASE = {METPLUS_BASE}/../METcalcpy


[plot_spectra]
# settings for plotting the cross-spectra

USER_SCRIPT_COMMAND = python {METPLUS_BASE}/parm/use_cases/model_applications/s2s/UserScript_
↪fcstS2S_obsERAI_CrossSpectra/cross_spectra_plot.py

LOG_FILE = "cross_spectra_plot.log"
LOG_LEVEL = "INFO"
```

MET Configuration

There are no MET tools used in this use case.

Python Embedding

There is no python embedding in this use case

Python Scripts

This use case uses a Python script to perform plotting

```
#!/usr/bin/env python3

"""
This is an example script for plotting cross spectral components. The script reads in output␣
↪files computed
by the example_cross_spectra.py script and uses the plotly plotting routines in spacetime_
↪plot.py to generate
a panel plot of coherence spectra.
"""
import numpy as np
import os
import sys
import xarray as xr

import metplotpy.contributed.spacetime_plot.spacetime_plot as stp
import metcalcpy.util.read_env_vars_in_config as readconfig


# Read in the YAML config file
# user can use their own, if none specified at the command line,
# use the "default" example YAML config file, spectra_plot_coh2.py
# Using a custom YAML reader so we can use environment variables
plot_config_file = os.getenv("PLOT_SPECTRA_YAML_CONFIG_NAME","spectra_plot.yaml")

config_dict = readconfig.parse_config(plot_config_file)

# Retrieve settings from config file
#pathdata is now set in the METplus conf file
#pathdata = config_dict['pathdata'][0]
plotpath = config_dict['plotpath'][0]
print("Output path ",plotpath)
model = config_dict['model']
var2 = config_dict['var2']
```

(continues on next page)

(continued from previous page)

```
var3 = config_dict['var3']

# plot layout parameters
flim = 0.5  # maximum frequency in cpd for plotting
nWavePlt = 20  # maximum wavenumber for plotting
contourmin = 0.1  # contour minimum
contourmax = 0.8  # contour maximum
contourspace = 0.1  # contour spacing
N = [1, 2]  # wave modes for plotting
source = ""
spd = 4

symmetry = "symm"      #("symm", "asymm", "latband")
#filenames = os.environ.get("INPUT_FILE_NAMES","ERAI_TRMM_P_symn,ERAI_P_D850_symn,ERAI_P_
↪D200_symn").split(",")
#vars1 = ['ERAI P', 'ERAI P', 'ERAI P']
#vars2 = ['TRMM', 'ERAI D850', 'ERAI D200']
filenames = os.environ.get("PLOT_SPECTRA_INPUT_FILE_NAMES","ERAI_P_D850_symn,ERAI_P_D200_
↪symn,ERAI_P_D200_symn").split(",")
vars1 = [model+' P',model+' P',model+' P']
vars2 = [model+' '+var2,model+' '+var3,model+' '+var3]
nplot = len(vars1)

npanel =3
for pp in np.arange(0, nplot, 1):

    # read data from file
    var1 = vars1[pp]
    var2 = vars2[pp]
    print("Filename ",filenames[pp])
    fin = xr.open_dataset(filenames[pp])
    STC = fin['STC'][:, :, :]
    wnum = fin['wnum']
    freq = fin['freq']

    #ifreq = np.where((freq[:] >= 0) & (freq[:] <= flim))
    #iwave = np.where(abs(wnum[:]) <= nWavePlt)

    STC[:, freq[:] == 0, :] = 0.
    STC = STC.sel(wnum=slice(-nWavePlt, nWavePlt))
    STC = STC.sel(freq=slice(0, flim))
    coh2 = np.squeeze(STC[4, :, :])
    phs1 = np.squeeze(STC[6, :, :])
    phs2 = np.squeeze(STC[7, :, :])
    phs1.where(coh2 <= contourmin, drop=True)
```

(continues on next page)

(continued from previous page)

```
    phs2.where(coh2 <= contourmin, drop=True)
    pow1 = np.squeeze(STC[0, :, :])
    pow2 = np.squeeze(STC[1, :, :])
    pow1.where(pow1 <= 0, drop=True)
    pow2.where(pow2 <= 0, drop=True)

    if pp == 0:
        ifreq = np.where((freq[:] >= 0) & (freq[:] <= flim))
        iwave = np.where(abs(wnum[:]) <= nWavePlt)
        Coh2 = np.full([npanel, len(freq[ifreq]), len(wnum[iwave])], np.nan)
        Phs1 = np.full([npanel, len(freq[ifreq]), len(wnum[iwave])], np.nan)
        Phs2 = np.full([npanel, len(freq[ifreq]), len(wnum[iwave])], np.nan)
        Pow1 = np.full([npanel, len(freq[ifreq]), len(wnum[iwave])], np.nan)
        Pow2 = np.full([npanel, len(freq[ifreq]), len(wnum[iwave])], np.nan)
        k = wnum[iwave]
        w = freq[ifreq]

    Coh2[pp, :, :] = coh2
    Phs1[pp, :, :] = phs1
    Phs2[pp, :, :] = phs2
    Pow1[pp, :, :] = np.log10(pow1)
    Pow2[pp, :, :] = np.log10(pow2)

    phstmp = Phs1
    phstmp = np.square(Phs1) + np.square(Phs2)
    phstmp = np.where(phstmp == 0, np.nan, phstmp)
    scl_one = np.sqrt(1 / phstmp)
    Phs1 = scl_one * Phs1
    Phs2 = scl_one * Phs2

# create output directory if it does not exist
if not os.path.exists(plotpath):
    print(f"Creating output directory: {plotpath}")
    os.makedirs(plotpath)

# plot coherence
stp.plot_coherence(Coh2, Phs1, Phs2, symmetry, source, vars1, vars2, plotpath, flim, 20,␣
↪contourmin, contourmax,
                   contourspace, npanel, N)

# check if output file exists since plotting function
# doesn't return an error code on failure
expected_file = os.path.join(plotpath,
                             'SpaceTimeCoherence_.png')
if not os.path.exists(expected_file):
```

(continues on next page)

(continued from previous page)

```
    print(f"ERROR: Could not create output file: {expected_file}")
    sys.exit(1)
```

Running METplus

This use case can be run two ways:

1) Passing in UserScript_fcstS2S_obsERAI_CrossSpectra.conf, then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s/UserScript_fcstS2S_
↪obsERAI_CrossSpectra.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_fcstS2S_obsERAI_CrossSpectra.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s/UserScript_
↪fcstS2S_obsERAI_CrossSpectra.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

  and for the [exe] section, you will need to define the location of NON-MET executables. If the executable is in the user's path, METplus will find it from the name. If the executable is not in the path, specify the full path to the executable here (i.e. RM = /bin/rm) The following executables are required for performing series analysis use cases:

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y

[exe]
RM = /path/to/rm
CUT = /path/to/cut
TR = /path/to/tr
NCAP2 = /path/to/ncap2
```

(continues on next page)

(continued from previous page)

```
CONVERT = /path/to/convert
NCDUMP = /path/to/ncdump
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Keywords

Note:

- UserScriptUseCase
- S2SAppUseCase
- METcalcpyUseCase
- METplotpyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s-UserScript_fcstS2S_obsERAI_CrossSpectra.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.10.2 Grid-Stat and Series-Analysis: BMKG APIK Seasonal Forecast

model_applications/s2s/GridStat_SeriesAnalysis _fcstNMME_obsCPC _seasonal_forecast.conf

Scientific Objective

The process of seasonal forecasting with a time horizon of one to many months (typically 6 to 9 months) poses new challenges to tools primarily developed for weather forecasting that cover a few days. These challenges include two aspects in particular: (1) a dramatically expanded time variable, and (2) a verification that is by design backward oriented using extensive hindcasts over past decades rather than the rapid verification possible in short-range weather forecasting. Therefore, the scientific objective of the seasonal forecast usecase involves the expansion of options to describe time as well as the strategic selection of hindcasts.

Time:

Commonly METplus expresses time intervals in the minutes, hours, and days. Month and year intervals were not supported since there is not a constant length for these units. Therefore, modifications to METplus were made to support these intervals by determining the offset relative to a given time.

Input data:

The input data from seasonal forecasts is generally based on daily, weekly, decadal (10-day), monthly or seasonal time integrated intervals. The time variable therefore is often no longer a simple snapshot of the system but rather representing an average, a sum (precipitation), or a particular statistic (maximum wind, minimum temperature, wind variability) over the integration time period. This requires some adjustment from the traditional approach in forecast verification where forecast time ("valid-time") is simply a snapshot out of a continuous run.

Hindcasts:

The objective of seasonal forecasts is no longer the exact location and intensity of one particular weather event, such as a storm, a frontal passage, or high wind conditions. Rather, seasonal forecasting focuses more on the statistical properties over a period of time, be it a 10-day interval, a month, or even a three month season. The verification of a new, forward looking seasonal forecast requires assessments of the forecast systems ability to appropriately forecast that longrange behavior of the weather (here, only atmospheric verification is considered, but the same concept would apply ocean or any other longrange forecast system). Because weather properties commonly change significantly over the course of the season, samples to verify the prognostic system can not be taken from the immediate days, weeks of months before the forecast. Hindcasting in the seasonal context requires a complete set of forecasts based on the same season but during past years. A current July-1 2019 forecast, therefore requires many July-1 forecasts for as many years in the past as possible, given that the forecast system is the same as the one used for the current forecast cycle into the future. Operational centers offer hindcasts, also sometimes called "re-forecasts", with the current, most up-to-date forecast system. MET and METplus therefore need to be able to extract the appropriate collection of past forecasts. This includes the identification of the same Julian-day-of-Year init-dates from forecasts cycles from past years, and then identify the different lead-times of interest generally ranging from one to 6 or more months.

Verification:

The verification steps can then utilize the existing collection of verification tools. In comparison to weather forecasts, the only difference is that the data, as stated above, are not snapshots but time-integrated values (averages, sums, statistics) that are representing a whole period of time. The verification then focuses on comparisons of these derivatives of the forecast simulations. In practice, a further step might be added prior to, or as a key step during verification: the formation of anomalies of the forecasts compared to long-term expected averages. A rainfall forecasts can therefore be verified in both absolute as well as anomaly context where some analyses might focus on extreme rainfall threshold exeedance of, for example, 500mm per month. At the same time, the same forecast might be verified for the 3 months rainfall average in comparison with the long-term expected mean. The verification might then assess how well the system can foresee the occurrence of below average rainfall over the season, and possibly some selected thresholds there (e.g., ability to forecast mean seasonal rainfall below the 10-th percentile of seasonal rainfall). Finally, flexibility in formulating forecast verification strategies is important as forecast skill might vary by location, the timing within the seasonal cycle, or the state of the evolving coupled system (the rapid onset of a strong El Nino will lead to significantly different forecast skill compared to a neutral state in the Pacific). Memory from past months, for example when considering accumulated soil moisture, might also influence the forecast skills. Seasonal forecast verification therefore requires understanding of the climate system; MET and METplus then need to offer the flexibility to tailor verification strategies and to potentially craft conditional approaches.

Overall, seasonal forecasts don't require a new verification approach. It does however put demands

on the flexibility of dealing with a significantly exapanded range of the time variable as well as logistic infrastructure to select appropriate hindcast samples from long hindcast or re-forecast archives. Scientifically, the challenges are mostly restricted in the appropriate formulation of verificaation questions that address specific forecast objectives. Compared to weather forecastsing, seasonal forecasts need to draw their skill from slowly changing components in the coupled Earth system while acknowledging the high-frequency noise of weather superposed on these 'climatologically' evolving background conditions. In many regions of the world, the noise might dominate that background climate and forecast skill is low. It is therefore the task of seasonal forecast verification to identify where there is actually skill for particular properties of the forecasts over a wide range of lead-times. The skill might be dependent on location, on the timing within the seasonal cycle, or even on the evolving state of the coupled system.

Datasets

All datasets are traditionally in netCDF format. Grids are either regular gaussian Latitude/Longitude grids or they are Lambert-conformal WRF grids.

The forecast datasets contain weekly, monthly or seasonally integrated data. Here, the time format of the use-case is monthly. Since the verification is done on the hindcasts rather than the forecast (would require another 6 months of waiting), the key identification here is the month of initialization and then the lead-time of the forecast of interest.

The hindcast data, the 'observational' data that is to be compared to the forecast, is a collection of datasets formatted in equivalent format to the forecast. The hindcast ensemble is identified through the year in the filename (as well as in the time variable inside the netCDF file).

Forecast Datasets:

NMME * variable of interest: pr (precipitation: cumulative monthly sum) * format of precipitation variable: time,lat,lon (here dimensions: 29,181,361) with time variable representing 29 samples of same Julian InitTime of hindcasts over past 29 years.

Hindcast Datasets:

Observational Dataset:

- CPC precipitation reference data (same format and grid)

METplus Components

This use case loops over initialization years and processes forecast lead months with GridStat It also processes the output of GridStat using two calls to SeriesAnalysis.

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- netCDF4

METplus Workflow

The following tools are used for each run time: GridStat

This example loops by initialization time. Each initialization time is July of each year from 1982 to 2010. For each init time it will run once, processing forecast leads 1 month through 5 months. The following times are processed:

Run times:

Init: 1982-07
Forecast leads: 1 month, 2 months, 3 months, 4 months, 5 months

Init: 1983-07
Forecast leads: 1 month, 2 months, 3 months, 4 months, 5 months

Init: 1984-07
Forecast leads: 1 month, 2 months, 3 months, 4 months, 5 months

Init: 1985-07
Forecast leads: 1 month, 2 months, 3 months, 4 months, 5 months

...

Init: 2009-07
Forecast leads: 1 month, 2 months, 3 months, 4 months, 5 months

Init: 2010-07
Forecast leads: 1 month, 2 months, 3 months, 4 months, 5 months

METplus Configuration

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s/GridStat_
↪SeriesAnalysis_fcstNMME_obsCPC_seasonal_forecast.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat, SeriesAnalysis(climo), SeriesAnalysis(full_stats)


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m
INIT_BEG = 198207
INIT_END = 201007
INIT_INCREMENT = 1Y

LEAD_SEQ = 1m, 2m, 3m, 4m, 5m, 6m

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_LEAD


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
```

(continues on next page)

(continued from previous page)

```
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/NMME/hindcast/monthly
FCST_GRID_STAT_INPUT_TEMPLATE = nmme_pr_hcst_{init?fmt=%b}IC_{valid?fmt=%m}_*.nc

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/NMME/obs
OBS_GRID_STAT_INPUT_TEMPLATE = obs_cpc_pp.1x1.nc

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/s2s/GridStat_SeriesAnalysis_fcstNMME_
↪obsCPC_seasonal_forecast/GridStat

BOTH_SERIES_ANALYSIS_INPUT_DIR = {GRID_STAT_OUTPUT_DIR}
BOTH_SERIES_ANALYSIS_INPUT_TEMPLATE = grid_stat_{MODEL}-hindcast_precip_vs_{OBTYPE}_IC{init?
↪fmt=%Y%b}_V{valid?fmt=%Y%m}01_*pairs.nc

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/s2s/GridStat_SeriesAnalysis_
↪fcstNMME_obsCPC_seasonal_forecast/SeriesAnalysis
SERIES_ANALYSIS_OUTPUT_TEMPLATE = series_analysis_{MODEL}_{OBTYPE}_stats_F{lead?fmt=%2m}_
↪{instance?fmt=%s}.nc

[full_stats]

SERIES_ANALYSIS_CLIMO_MEAN_INPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_TEMPLATE = series_analysis_{MODEL}_{OBTYPE}_stats_F{lead?
↪fmt=%2m}_climo.nc

[config]


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = NMME
OBTYPE = CPC

FCST_GRID_STAT_VAR1_NAME = pr
FCST_GRID_STAT_VAR1_LEVELS = "({valid?fmt=%Y%m01_000000},*,*)"
FCST_GRID_STAT_VAR1_THRESH = >0, >50, >100, >150, >200, >250, >300, >400, >500

OBS_GRID_STAT_VAR1_NAME = precip
OBS_GRID_STAT_VAR1_LEVELS = "({valid?fmt=%Y%m01_000000},*,*)"
OBS_GRID_STAT_VAR1_THRESH = >0, >50, >100, >150, >200, >250, >300, >400, >500
```

(continues on next page)

(continued from previous page)

```
FCST_SERIES_ANALYSIS_VAR1_NAME = FCST_precip_FULL
FCST_SERIES_ANALYSIS_VAR1_LEVELS = "(*,*)"

OBS_SERIES_ANALYSIS_VAR1_NAME = OBS_precip_FULL
OBS_SERIES_ANALYSIS_VAR1_LEVELS = "(*,*)"


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CNT = STAT
GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT

GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE

GRID_STAT_NC_PAIRS_VAR_NAME = precip

GRID_STAT_OUTPUT_PREFIX = {MODEL}-hindcast_{CURRENT_OBS_NAME}_vs_{OBTYPE}_IC{init?fmt=%Y%b}_V
↪{valid?fmt=%Y%m%d}

###
# SeriesAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#seriesanalysis
###

SERIES_ANALYSIS_DESC = hindcast

SERIES_ANALYSIS_CAT_THRESH = >=50, >=100, >=150, >=200, >=250, >=300, >=400, >=500

SERIES_ANALYSIS_VLD_THRESH = 0.50

SERIES_ANALYSIS_BLOCK_SIZE = 360*181

SERIES_ANALYSIS_IS_PAIRED = False

SERIES_ANALYSIS_GENERATE_PLOTS = no
SERIES_ANALYSIS_GENERATE_ANIMATIONS = no

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False


SERIES_ANALYSIS_STAT_LIST = OBAR
```

(continues on next page)

(continued from previous page)

```
[full_stats]

SERIES_ANALYSIS_STAT_LIST =TOTAL, FBAR, OBAR, ME, MAE, RMSE, ANOM_CORR, PR_CORR
SERIES_ANALYSIS_CTS_LIST = BASER, CSI, GSS
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

GridStatConfig_wrapped

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
```

(continues on next page)

(continued from previous page)

```
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
```

(continues on next page)

(continued from previous page)

```
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
```

(continues on next page)

(continued from previous page)

```
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

SeriesAnalysisConfig_wrapped

Note: See the *SeriesAnalysis MET Configuration* (page 232) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Series-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
//
//desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
//obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

censor_thresh = [];
censor_val    = [];
//cat_thresh =
${METPLUS_CAT_THRESH}
cnt_thresh    = [ NA ];
cnt_logic     = UNION;

//
// Forecast and observation fields to be verified
```

(continues on next page)

(continued from previous page)

```
//
fcst = {
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_CAT_THRESH}
   ${METPLUS_FCST_FIELD}
}
obs = {
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

//
// Number of grid points to be processed concurrently.  Set smaller to use
// less memory but increase the number of passes through the data.
//
//block_size =
${METPLUS_BLOCK_SIZE}

//
// Ratio of valid matched pairs to compute statistics for a grid point
//
//vld_thresh =
${METPLUS_VLD_THRESH}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_stats = {
${METPLUS_OUTPUT_STATS_DICT}

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_SeriesAnalysis_fcstNMME_obsCPC_seasonal_forecast.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s/GridStat_
↪SeriesAnalysis_fcstNMME_obsCPC_seasonal_forecast.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_SeriesAnalysis_fcstNMME_obsCPC_seasonal_forecast.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s/GridStat_
↪SeriesAnalysis_fcstNMME_obsCPC_seasonal_forecast.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/s2s/GridStat_SeriesAnalysis_fcstNMME_obsCPC_seasonal_forecast/GridStat (relative to **OUTPUT_BASE**)

For each month and year there will be two files written:

```
* grid_stat_NMME-hindcast_precip_vs_CPC_IC{%Y%b}01_2301360000L_20081001_000000V.stat
* grid_stat_NMME-hindcast_precip_vs_CPC_IC{%Y%b}01_2301360000L_20081001_000000V_pairs.nc
```

Output from SeriesAnalysis will be found in model_applications/s2s/GridStat_SeriesAnalysis_fcstNMME_obsCPC_seasc
(relative to **OUTPUT_BASE**)

For each month there will be two files written:

```
* series_analysis_NMME_CPC_stats_ICJul_{%m}_climo.nc
* series_analysis_NMME_CPC_stats_ICJul_{%m}_full_stats.nc
```

Keywords

Note:

- GridStatToolUseCase
- SeriesAnalysisUseCase
- NetCDFFileUseCase
- LoopByMonthFeatureUseCase
- NCAROrgUseCase
- RuntimeFreqUseCase
- ClimatologyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s-GridStat_SeriesAnalysis_fcstNMME_obsCPC_seasonal_forecast.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.10.3 TCGen: Genesis Density Function (GDF) and Track Density Function (TDF)

model_applications/s2s/TCGen_fcstGFSO_obsBDECKS_GDF_TDF.conf

Scientific Objective

Tropocal cyclone (TC) genesis density function (GDF) and track density function (TDF) are designed to quantitatively evaluate geographic distributions of TC activities including TC genesis frequency and subsequent TC tracks. Spatial patterns of long-term averaged GDF or TDF on the regional or global scale are particularly useful to evaluate TC forecasts against those derived from an observational best-track dataset, such as IBTrACS or ATCF B-decks, from a climate perspective. The metrics can help assess the forecast biases (under- or over-prediction) of TC formations or TC vortices around particular locations in a numerical model.

For demonstration purposes, only cyclone tracker output and b-decks data for 2016 are used.

The following settings are used in the use case, all of which are configurable in the METplus configuration file (see below).

Forecast genesis event criteria:

Minimum forecast lead: 48h
Maximum forecast lead: 120h
Maximum velocity threshold: >= 16.5 m/s
Minimum TC duration: 24h

Observed genesis event criteria:

Minimum TC duration: 24h
Maximum velocity threshold: >= 17.0 m/s
Minimum TC Category: TD

Matching settings:

Genesis matching window: +/- 24h
Early genesis matching window: -120h
Late genesis matching window: +120h
Genesis hit scoring window: +/- 24h
Early genesis hit scoring window: -120h
Late genesis hit scoring window: +120h
Matching and Scoring radius: 555 km

In addition to the above settings, normalization is performed on the metrics by the number of years included in the dataset (in this example, just one), and the total number of model forecasts valid at the time of an observed genesis event. The latter can also be thought of as the total number of chances that the model had to forecast a genesis event.

Datasets

Both forecast and observation datasets for this use case must adhere to the ATCF format.

Forecast data: GFDL Cyclone Tracker output configured for "genesis mode" for the FV3GFS model. This configuration used an experimental GFSv15 physics package, and had a horizontal grid spacing of ~25 km with 64 vertical levels.

Observation data: Global ATCF B-decks files from the National Hurricane Center (NHC) and Joint Typhoon Warning Center (JTWC)

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

The MET TCGen tool requires forecast data to be provided from the GFDL cyclone tracker. More information about the GFDL cyclone tracker can be found here: https://dtcenter.org/community-code/gfdl-vortex-tracker

Archives of ATCF B-decks files can be found at these locations:

https://www.metoc.navy.mil/jtwc/jtwc.html?best-tracks
https://www.nhc.noaa.gov/data/#hurdat

Software Versions

This use case was developed with specific versions of various software and Python packages. Any deviation from these versions may require re-configuration or adaptation to reproduce the results shown.

Names and version numbers:

```
python-3.6.3
cartopy-0.18.0
matplotlib-3.1.2
MET-10.0.0
METplus-4.0.0
METplotpy-1.0.0
```

METplus Components

This use case utilizes the MET TCGen tool to generate matched pairs of TC genesis, and then uses Python Embedding to compute the TDF and GDF metrics and create graphics for the year 2016.

METplus Workflow

The following tools are used for each run time: TCGen, Python

The TCGen tool is designed to be provided a single file pair or a directory containing a list of files, rather than loop over valid or initialization times. Thus, a single year is used in the METplus configuration file and wildcard symbols are provided to gather all the tracker and genesis input files at each input directory.

METplus Configuration

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s/TCGen_fcstGFSO_
↪obsBDECKS_GDF_TDF.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCGen, UserScript


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
```

(continues on next page)

(continued from previous page)

```
INIT_TIME_FMT = %Y
INIT_BEG = 2016

LEAD_SEQ =

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_FOR_EACH


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

TC_GEN_TRACK_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/TCGen_fcstGFSO_obsBDECKS_GDF_
↪TDF/obs/bdecks/{INIT_BEG}
TC_GEN_TRACK_INPUT_TEMPLATE = *.dat

TC_GEN_GENESIS_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/TCGen_fcstGFSO_obsBDECKS_GDF_
↪TDF/fcst/tracker/reformat/{INIT_BEG}
TC_GEN_GENESIS_INPUT_TEMPLATE = *.fort.66

TC_GEN_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/s2s/TCGen_fcstGFSO_obsBDECKS_GDF_TDF/
↪TCGen
TC_GEN_OUTPUT_TEMPLATE = tc_gen_{init?fmt=%Y}


###
# TCGen Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcgen
###

TC_GEN_INIT_FREQ = 6

TC_GEN_VALID_FREQ = 6

TC_GEN_FCST_HR_WINDOW_BEGIN = 48

TC_GEN_FCST_HR_WINDOW_END = 120

TC_GEN_MIN_DURATION = 24

TC_GEN_FCST_GENESIS_VMAX_THRESH = >=16.5
TC_GEN_FCST_GENESIS_MSLP_THRESH = NA
```

(continues on next page)

(continued from previous page)

```
TC_GEN_BEST_GENESIS_TECHNIQUE = BEST
TC_GEN_BEST_GENESIS_CATEGORY = TD
TC_GEN_BEST_GENESIS_VMAX_THRESH = >=17.0
TC_GEN_BEST_GENESIS_MSLP_THRESH = NA

TC_GEN_OPER_TECHNIQUE =

TC_GEN_FILTER_MODEL = GFSO
TC_GEN_GDF_FILTER_DESC = GDF
TC_GEN_EARLY_FILTER_DESC = GDF_EARLY
TC_GEN_LATE_FILTER_DESC = GDF_LATE

TC_GEN_FILTER_1 = model = "{TC_GEN_FILTER_MODEL}"; desc = "{TC_GEN_GDF_FILTER_DESC}"; dev_
↪hit_window = { beg = -24; end = 24; }; dev_hit_radius = 555; genesis_match_window = { beg␣
↪= -24; end = 24;};
TC_GEN_FILTER_2 = model = "{TC_GEN_FILTER_MODEL}"; desc = "{TC_GEN_EARLY_FILTER_DESC}"; dev_
↪hit_window = { beg = -120; end = 0; }; dev_hit_radius = 555; genesis_match_window = { beg␣
↪= -120; end = 0;};
TC_GEN_FILTER_3 = model = "{TC_GEN_FILTER_MODEL}"; desc = "{TC_GEN_LATE_FILTER_DESC}"; dev_
↪hit_window = { beg = 0; end = 120; }; dev_hit_radius = 555; genesis_match_window = { beg =␣
↪0; end = 120;};

TC_GEN_DESC = ALL

TC_GEN_DLAND_THRESH = NA

TC_GEN_GENESIS_MATCH_RADIUS = 555

TC_GEN_GENESIS_MATCH_POINT_TO_TRACK = False

TC_GEN_GENESIS_MATCH_WINDOW_BEG = 0
TC_GEN_GENESIS_MATCH_WINDOW_END = 0

TC_GEN_OPS_HIT_WINDOW_BEG = 0
TC_GEN_OPS_HIT_WINDOW_END = 48

TC_GEN_DEV_HIT_RADIUS = 500

TC_GEN_DEV_HIT_WINDOW_BEGIN = -24
TC_GEN_DEV_HIT_WINDOW_END = 24

TC_GEN_OPS_HIT_TDIFF = 48

TC_GEN_DISCARD_INIT_POST_GENESIS_FLAG = True
```

(continues on next page)

(continued from previous page)

```
TC_GEN_DEV_METHOD_FLAG = True

TC_GEN_OPS_METHOD_FLAG = False

TC_GEN_CI_ALPHA = 0.05

TC_GEN_OUTPUT_FLAG_FHO = NONE
TC_GEN_OUTPUT_FLAG_CTC = BOTH
TC_GEN_OUTPUT_FLAG_CTS = BOTH
TC_GEN_OUTPUT_FLAG_GENMPR = BOTH

TC_GEN_NC_PAIRS_FLAG_LATLON = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_GENESIS = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_TRACKS  = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_FY_OY   = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_FY_ON   = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_GENESIS = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_TRACKS  = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_FY_OY   = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_FN_OY   = TRUE

TC_GEN_VALID_MINUS_GENESIS_DIFF_THRESH = >0

TC_GEN_BEST_UNIQUE_FLAG = TRUE

TC_GEN_DLAND_FILE = MET_BASE/tc_data/dland_global_tenth_degree.nc

TC_GEN_BASIN_FILE = MET_BASE/tc_data/basin_global_tenth_degree.nc

TC_GEN_NC_PAIRS_GRID = G003


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

USER_SCRIPT_INPUT_TEMPLATE = {TC_GEN_OUTPUT_DIR}/tc_gen_{init?fmt=%Y}_pairs.nc
SCRIPT_DIR = {PARM_BASE}/use_cases/model_applications/s2s/TCGen_fcstGFSO_obsBDECKS_GDF_TDF
USER_SCRIPT_COMMAND = {SCRIPT_DIR}/UserScript_fcstGFSO_obsBDECKS_GDF_TDF.py {USER_SCRIPT_
↪INPUT_TEMPLATE}

[user_env_vars]
TCGEN_INIT_FREQ = {TC_GEN_INIT_FREQ}
TCGEN_MIN_LEAD = {TC_GEN_FCST_HR_WINDOW_BEGIN}
```

(continues on next page)

(continued from previous page)

```
TCGEN_MAX_LEAD = {TC_GEN_FCST_HR_WINDOW_END}
GDF_LAT_HALF_DELTA = 5.0
GDF_LON_HALF_DELTA = 5.0
GDF_NORM_YEARS = 1.0
GDF_PLOT_OUTDIR = {OUTPUT_BASE}/images
GDF_MODEL_STRING = {TC_GEN_FILTER_MODEL}
GDF_OBS_STRING = BEST
GDF_DESC_STRING = {TC_GEN_GDF_FILTER_DESC}
GDF_EARLY_STRING = {TC_GEN_EARLY_FILTER_DESC}
GDF_LATE_STRING = {TC_GEN_LATE_FILTER_DESC}
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

TCGenConfig_wrapped

Note: See the *TCGen MET Configuration* (page 266) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// TC-Gen configuration file.
//
// For additional information, see the MET_BASE/config/README_TC file.
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

////////////////////////////////////////////////////////////////////////////////
//
// Genesis event definition criteria.
//
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Model initialization frequency in hours, starting at 0.
//
// init_freq =
${METPLUS_INIT_FREQ}

//
// Valid hour frequency to be analyzed in hours, starting at 0
//
// valid_freq =
${METPLUS_VALID_FREQ}

//
// Forecast hours to be searched for genesis events
//
// fcst_hr_window =
${METPLUS_FCST_HR_WINDOW_DICT}

//
// Minimum track duration for genesis event in hours.
//
// min_duration =
${METPLUS_MIN_DURATION}

//
// Forecast genesis event criteria.  Defined as tracks reaching the specified
// intensity category, maximum wind speed threshold, and minimum sea-level
// pressure threshold.  The forecast genesis time is the valid time of the first
// track point where all of these criteria are met.
//
// fcst_genesis =
${METPLUS_FCST_GENESIS_DICT}

//
// BEST track genesis event criteria.  Defined as tracks reaching the specified
// intensity category, maximum wind speed threshold, and minimum sea-level
// pressure threshold.  The BEST track genesis time is the valid time of the
// first track point where all of these criteria are met.
//
// best_genesis =
${METPLUS_BEST_GENESIS_DICT}

//
```

(continues on next page)

(continued from previous page)

```
// Operational track technique name
//
// oper_technique =
${METPLUS_OPER_TECHNIQUE}

////////////////////////////////////////////////////////////////////////////////
//
// Track filtering options
// May be specified separately in each filter array entry.
//
////////////////////////////////////////////////////////////////////////////////

//
// Array of dictionaries containing the track filtering options
// If empty, a single filter is defined using the top-level settings.
//
// filter =
${METPLUS_FILTER}

//
// Description written to output DESC column
//
// desc =
${METPLUS_DESC}

//
// Forecast ATCF ID's
// If empty, all ATCF ID's found will be processed.
// Statistics will be generated separately for each ATCF ID.
//
// model =
${METPLUS_MODEL}

//
// BEST and operational track storm identifiers
//
// storm_id =
${METPLUS_STORM_ID}

//
// BEST and operational track storm names
//
// storm_name =
${METPLUS_STORM_NAME}
```

(continues on next page)

(continued from previous page)

```
//
// Forecast and operational initialization times to include or exclude
//
// init_beg =
${METPLUS_INIT_BEG}

// init_end =
${METPLUS_INIT_END}

// init_inc =
${METPLUS_INIT_INC}

// init_exc =
${METPLUS_INIT_EXC}

//
// Forecast, BEST, and operational valid time window
//
// valid_beg =
${METPLUS_VALID_BEG}

// valid_end =
${METPLUS_VALID_END}

//
// Forecast and operational initialization hours
//
// init_hour =
${METPLUS_INIT_HOUR}

//
// Forecast and operational lead times in hours
//
// lead =
${METPLUS_LEAD}

//
// Spatial masking region (path to gridded data file or polyline file)
//
// vx_mask =
${METPLUS_VX_MASK}

//
// Spatial masking of hurricane basin names from the basin_file
//
```

(continues on next page)

(continued from previous page)

```
// basin_mask =
${METPLUS_BASIN_MASK}

//
// Distance to land threshold
//
//dland_thresh =
${METPLUS_DLAND_THRESH}

////////////////////////////////////////////////////////////////////////////////
//
// Matching and scoring options
// May be specified separately in each filter array entry.
//
////////////////////////////////////////////////////////////////////////////////

//
// Genesis matching logic. Compare the forecast genesis point to all points in
// the Best track (TRUE) or the single Best track genesis point (FALSE).
//
//genesis_match_point_to_track =
${METPLUS_GENESIS_MATCH_POINT_TO_TRACK}

//
// Radius in km to search for a matching genesis event
//
// genesis_match_radius =
${METPLUS_GENESIS_MATCH_RADIUS}

//
// Time window in hours, relative to the model genesis time, to search for a
// matching Best track point
//
//genesis_match_window = {
${METPLUS_GENESIS_MATCH_WINDOW_DICT}

//
// Radius in km for a development scoring method hit
//
// dev_hit_radius =
${METPLUS_DEV_HIT_RADIUS}

//
// Time window in hours for a development scoring method hit
//
```

(continues on next page)

(continued from previous page)

```
// dev_hit_window =
${METPLUS_DEV_HIT_WINDOW_DICT}

// Time window in hours for the Best track genesis minus model initialization
// time difference for an operational scoring method hit
//
//ops_hit_window = {
${METPLUS_OPS_HIT_WINDOW_DICT}

//
// Discard genesis forecasts for initializations at or after the matching
// BEST track genesis time
//
// discard_init_post_genesis_flag =
${METPLUS_DISCARD_INIT_POST_GENESIS_FLAG}

//
// Scoring methods to be applied
//
//dev_method_flag =
${METPLUS_DEV_METHOD_FLAG}

// ops_method_flag =
${METPLUS_OPS_METHOD_FLAG}

////////////////////////////////////////////////////////////////////////////////
//
// Output options
// May be specified separately in each filter array entry.
//
////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval alpha value
//
// ci_alpha =
${METPLUS_CI_ALPHA}

//
// Statistical output types
//
// output_flag =
${METPLUS_OUTPUT_FLAG_DICT}

//
```

(continues on next page)

(continued from previous page)

```
// NetCDF genesis pair counts
//
// nc_pairs_flag =
${METPLUS_NC_PAIRS_FLAG_DICT}

//
// Specify which track points should be counted by thresholding the track point
// valid time minus genesis time difference.
//
// valid_minus_genesis_diff_thresh =
${METPLUS_VALID_MINUS_GENESIS_DIFF_THRESH}

//
// Count unique BEST track genesis event locations (TRUE) versus counting the
// location for all pairs (FALSE).
//
// best_unique_flag =
${METPLUS_BEST_UNIQUE_FLAG}

////////////////////////////////////////////////////////////////////////////////
//
// Global settings
// May only be specified once.
//
////////////////////////////////////////////////////////////////////////////////

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
// dland_file =
${METPLUS_DLAND_FILE}

//
// Specify the NetCDF file containing a gridded representation of the
// global basins.
//
// basin_file =
${METPLUS_BASIN_FILE}

//
// NetCDF genesis pairs grid
//
// nc_pairs_grid =
${METPLUS_NC_PAIRS_GRID}
```

(continues on next page)

(continued from previous page)

```
//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V10.0.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses a Python embedding script to create output graphics

parm/use_cases/model_applications/s2s/TCGen_fcstGFSO_obsBDECKS_GDF_TDF/UserScript_fcstGFSO_obsBDECKS_

```
#!/usr/bin/env python3

"""UserScript to compute density variables for the METplus S2S TDF/GDF use case

This script is used to read in netCDF output from the MET TCGen tool and compute
various density variables related to the Genesis Density Function (GDF) and
Track Density Function (TDF) metrics for subseasonal-to-seasonal applications.

Contains the following functions:
  * as_density()

Author: Daniel R. Adriaansen
Date: 24 March 2021

"""

import xarray as xr
import os
import sys
import datetime
import multiprocessing

# Import METplotpy
from metplotpy.contributed.tc_s2s_panel import plot_tc_s2s_panel as tc_s2s_panel

# Enviromnment variables for use case
GDF_INPUT_FILENAME = sys.argv[1]
GDF_LAT_HALF_DELTA = float(str(os.environ.get('GDF_LAT_HALF_DELTA',5.0)))
```

(continues on next page)

(continued from previous page)

```
GDF_LON_HALF_DELTA = float(str(os.environ.get('GDF_LON_HALF_DELTA',5.0)))
GDF_MODEL_STRING = str(os.environ.get('GDF_MODEL_STRING','TESTMODEL'))
GDF_OBS_STRING = str(os.environ.get('GDF_OBS_STRING','TESTOBS'))
GDF_DESC_STRING = str(os.environ.get('GDF_DESC_STRING','GDF'))
GDF_EARLY_STRING = str(os.environ.get('GDF_EARLY_STRING','GDF_EARLY'))
GDF_LATE_STRING = str(os.environ.get('GDF_LATE_STRING','GDF_LATE'))
GDF_NORM_YEARS = float(str(os.environ.get('GDF_NORM_YEARS',1.0)))

# Compute the total number of model forecasts that could have forecasted a hypothetical␣
↪genesis event
# within the user defined lead window
lead_step = int(str(os.environ.get('TCGEN_INIT_FREQ')))
shortest_lead = int(str(os.environ.get('TCGEN_MIN_LEAD')))
longest_lead = int(str(os.environ.get('TCGEN_MAX_LEAD')))
num_forecasts = float(len([shortest_lead + x for x in range(0,longest_lead,lead_step) if␣
↪shortest_lead+x <= longest_lead]))

# Local variables
DEBUG = False

# Create some netCDF varname strings to use when referencing the data
fcstgenvarname = f"{GDF_DESC_STRING}_{GDF_MODEL_STRING}_GENESIS"
fcsthitvarname = f"{GDF_DESC_STRING}_{GDF_MODEL_STRING}_DEV_FY_OY"
fcstfalmvarname = f"{GDF_DESC_STRING}_{GDF_MODEL_STRING}_DEV_FY_ON"
obsmissvarname = f"{GDF_DESC_STRING}_{GDF_MODEL_STRING}_BEST_DEV_FN_OY"
obsgenvarname = f"{GDF_DESC_STRING}_BEST_GENESIS"
fcsttrackvarname = f"{GDF_DESC_STRING}_{GDF_MODEL_STRING}_TRACKS"
obstrackvarname = f"{GDF_DESC_STRING}_BEST_TRACKS"
fcstearlygenvarname = f"{GDF_EARLY_STRING}_{GDF_MODEL_STRING}_GENESIS"
fcstearlyhitvarname = f"{GDF_EARLY_STRING}_{GDF_MODEL_STRING}_DEV_FY_OY"
fcstlategenvarname = f"{GDF_LATE_STRING}_{GDF_MODEL_STRING}_GENESIS"
fcstlatehitvarname = f"{GDF_LATE_STRING}_{GDF_MODEL_STRING}_DEV_FY_OY"
if DEBUG:
  print("\nUSING VARIABLE NAMES:")
  print(f"Forecast genesis events varname: {fcstgenvarname}")
  print(f"Hits (fy_oy) varname::           {fcsthitvarname}")
  print(f"False alarms (fy_on) varname:    {fcstfalmvarname}")
  print(f"Miss (fn_oy) varname:            {obsmissvarname}")
  print(f"Observed genesis events varname: {obsgenvarname}")
  print(f"Forecast track points varname:   {fcsttrackvarname}")
  print(f"Observed track points varname:   {obstrackvarname}")
  print(f"Early genesis event varname:     {fcstearlygenvarname}")
  print(f"Early genesis hit varname:       {fcstearlyhitvarname}")
  print(f"Late genesis event varname:      {fcstlategenvarname}")
  print(f"Late genesis hit varname:        {fcstlatehitvarname}")
```

(continues on next page)

(continued from previous page)

```
# Open the TCGen output file
tcgendata = xr.open_dataset(GDF_INPUT_FILENAME)

# Create 1D data to find locations of events
tcgendata1d = tcgendata.stack(adim=('lat','lon'))

# Get the lat/lon of EARLY FCST genesis events
earl_lat = tcgendata1d[fcstearlygenvarname].where(tcgendata1d[fcstearlygenvarname]>0.0,
↪drop=True)['lat'].values
earl_lon = tcgendata1d[fcstearlygenvarname].where(tcgendata1d[fcstearlygenvarname]>0.0,
↪drop=True)['lon'].values

# Get the lat/lon of LATE FCST genesis events
late_lat = tcgendata1d[fcstlategenvarname].where(tcgendata1d[fcstlategenvarname]>0.0,
↪drop=True)['lat'].values
late_lon = tcgendata1d[fcstlategenvarname].where(tcgendata1d[fcstlategenvarname]>0.0,
↪drop=True)['lon'].values

# Get the lat/lon of FCST genesis events
fcst_lat = tcgendata1d[fcstgenvarname].where(tcgendata1d[fcstgenvarname]>0.0,drop=True)['lat
↪'].values
fcst_lon = tcgendata1d[fcstgenvarname].where(tcgendata1d[fcstgenvarname]>0.0,drop=True)['lon
↪'].values

# Get the lat/lon of the OBS (BEST) genesis events
obs_lat = tcgendata1d[obsgenvarname].where(tcgendata1d[obsgenvarname]>0.0,drop=True)['lat'].
↪values
obs_lon = tcgendata1d[obsgenvarname].where(tcgendata1d[obsgenvarname]>0.0,drop=True)['lon'].
↪values

# Get the lat/lon of the FCST track points (based on genesis)
ftrk_lat = tcgendata1d[fcsttrackvarname].where(tcgendata1d[fcsttrackvarname]>0.0,drop=True)[
↪'lat'].values
ftrk_lon = tcgendata1d[fcsttrackvarname].where(tcgendata1d[fcsttrackvarname]>0.0,drop=True)[
↪'lon'].values

# Get the lat/lon of the OBS (BEST) track points (based on genesis)
otrk_lat = tcgendata1d[obstrackvarname].where(tcgendata1d[obstrackvarname]>0.0,drop=True)[
↪'lat'].values
otrk_lon = tcgendata1d[obstrackvarname].where(tcgendata1d[obstrackvarname]>0.0,drop=True)[
↪'lon'].values

# Function to take gridded counts of data and create a density plot given a lat/lon region␣
↪defined by GDF_LAT/LON_HALF_DELTA around these counts.
```

(continues on next page)

(continued from previous page)

```
# Input are the individual lats/lons of each location where there are any events, as well as␣
↪the actual gridded variable of counts
def as_density(elats,elons,grid_var,fcst):

  """Computes the density of an event based on a gridded count variable and the lat/lon of␣
↪each event

  Parameters
  ----------
  elats: list of floating point latitude values of individual events
  elons: list of floating point longitude values of individual events
  grid_var: Xarray DataArray containing the count at each event location
  fcst: a boolean to denote whether the grid_var is a forecast (True) or observation (False)␣
↪variable

  Returns
  -------
  Xarray DataArray object the with the same likeness as grid_var

  """

  # Create a DataArray that looks like the input grid_var
  dens_var = xr.zeros_like(grid_var,dtype='float32')

  # Try to re-write the while loop as a for loop
  #for clat,clon in tuple(zip(elats,elons)):
  llcnt = 0
  while llcnt < len(elats):

    clat = elats[llcnt]
    clon = elons[llcnt]

    # Latitude and longitude of subdomain around the point lat/lon
    glat = grid_var.lat[(grid_var.lat>=clat-GDF_LAT_HALF_DELTA) & (grid_var.lat<=clat+GDF_
↪LAT_HALF_DELTA)]
    glon = grid_var.lon[(grid_var.lon>=clon-GDF_LON_HALF_DELTA) & (grid_var.lon<=clon+GDF_
↪LON_HALF_DELTA)]

    # Get the number of events at the current point lat/lon
    nevent = grid_var.sel(lat=clat,lon=clon).values

    # Increment the dens_var where we want
    dens_var.loc[dict(lat=glat,lon=glon)] += nevent

    llcnt += 1
```

(continues on next page)

(continued from previous page)

```
  # Return the dens_var
  if fcst:
    return(dens_var/(GDF_NORM_YEARS*num_forecasts))
  else:
    return(dens_var/(GDF_NORM_YEARS))

# Create some lists of function arguments to as_density() to run in parallel
varlist = [fcstgenvarname,fcsthitvarname,fcstfalmvarname,fcsttrackvarname,obsgenvarname,
↪obsmissvarname,obstrackvarname,fcstlatehitvarname,fcstearlyhitvarname]
varlats = [fcst_lat,fcst_lat,fcst_lat,ftrk_lat,obs_lat,obs_lat,otrk_lat,late_lat,earl_lat]
varlons = [fcst_lon,fcst_lon,fcst_lon,ftrk_lon,obs_lon,obs_lon,otrk_lon,late_lon,earl_lon]
fcstobs = [True,True,True,True,False,True,False,True,True]
denvars = ['FCST_DENS','FYOY_DENS','FYON_DENS','FTRK_DENS','OBS_DENS','FNOY_DENS','OTRK_DENS
↪','LHIT_DENS','EHIT_DENS']

# Use multiprocessing to run in parallel
# Results is a list of DataArray objects
mp = multiprocessing.Pool(max(multiprocessing.cpu_count()-2, 1))
results = mp.starmap(as_density,[(x,y,tcgendata[z],f) for x,y,z,f  in tuple(zip(varlats,
↪varlons,varlist,fcstobs))])

# Unpack the results
for r,n in tuple(zip(results,denvars)):
  tcgendata[n] = r

if DEBUG:
  print("\nOBS_DENS")
  print(tcgendata['OBS_DENS'].min().values)
  print(tcgendata['OBS_DENS'].max().values)
  print("\nFCST_DENS")
  print(tcgendata['FCST_DENS'].min().values)
  print(tcgendata['FCST_DENS'].max().values)
  print("\nFYOY_DENS")
  print(tcgendata['FYOY_DENS'].min().values)
  print(tcgendata['FYOY_DENS'].max().values)
  print("\nFYON_DENS")
  print(tcgendata['FYON_DENS'].min().values)
  print(tcgendata['FYON_DENS'].max().values)
  print("\nFNOY_DENS")
  print(tcgendata['FNOY_DENS'].min().values)
  print(tcgendata['FNOY_DENS'].max().values)
  print("\nFTRK_DENS")
  print(tcgendata['FTRK_DENS'].min().values)
  print(tcgendata['FTRK_DENS'].max().values)
```

(continues on next page)

(continued from previous page)

```
  print("\nOTRK_DENS")
  print(tcgendata['OTRK_DENS'].min().values)
  print(tcgendata['OTRK_DENS'].max().values)
  print("\nEHIT_DENS")
  print(tcgendata['EHIT_DENS'].min().values)
  print(tcgendata['EHIT_DENS'].max().values)
  print("\nLHIT_DENS")
  print(tcgendata['LHIT_DENS'].min().values)
  print(tcgendata['LHIT_DENS'].max().values)

# Call plotting for GDF. tc_s2s_panel.plot_gdf() requires just the Xarray Dataset object
# Panel order for GDF is:
# 1. Total BEST (observed) genesis density
# 2. Total MODEL (forecast) genesis density
# 3. Difference 2-1
gdf_varlist = ['OBS_DENS','FCST_DENS']
tc_s2s_panel.plot_gdf(tcgendata[gdf_varlist],os.environ.get('GDF_PLOT_OUTDIR'))

# Call plotting for TDF. tc_s2s_panel.plot_tdf() requires just the Xarray Dataset object
# Panel order for TDF is:
# 1. Total BEST (observed) track points
# 2. Total FCST (hour 24-120) track points
# 3. FCST-BEST
tdf_varlist = ['FTRK_DENS','OTRK_DENS']
tc_s2s_panel.plot_tdf(tcgendata[tdf_varlist],os.environ.get('GDF_PLOT_OUTDIR'))

# Call plotting for GDF category. tc_s2s_panel.plot_gdf_cat() requires just the Xarray␣
↪Dataset object
# Panel order for GDF category is:
# 1. Total HITS density
# 2. Total EARLY HITS density
# 3. Total LATE HITS density
# 4. Total FALSE_ALARMS density
gdf_cat_varlist = ['FYOY_DENS','EHIT_DENS','LHIT_DENS','FYON_DENS']
tc_s2s_panel.plot_gdf_cat(tcgendata[gdf_cat_varlist],os.environ.get('GDF_PLOT_OUTDIR'))

# Call plotting for GDF UFS. tc_s2s_panel.plot_gdf_ufs() requires just the Xarray Dataset␣
↪object
# Panel order for GDF UFS is:
# 1. Total BEST (observed) genesis density (scatter is BEST genesis locations)
# 2. Total HITS density (scatter is BEST genesis locations)
# 3. Total FALSE_ALARMS density (scatter is FCST genesis locations, but only false?)
# 4. Total HITS+FALSE_ALARMS density (scatter is FCST genesis locations, but only hits+false?
↪)
gdf_ufs_varlist = ['OBS_DENS','FYOY_DENS','FYON_DENS']
tc_s2s_panel.plot_gdf_ufs(tcgendata[gdf_ufs_varlist],os.environ.get('GDF_PLOT_OUTDIR'))
```

Running METplus

This use case can be run two ways:

1) Passing in TCGen_fcstGFSO_obsBDECKS_GDF_TDF.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s/TCGen_fcstGFSO_
↪obsBDECKS_GDF_TDF.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in TCGen_fcstGFSO_obsBDECKS_GDF_TDF.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s/TCGen_fcstGFSO_
↪obsBDECKS_GDF_TDF.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[config]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated.

Output from TCGen for this use case will be found in model_applications/s2s/TCGen_fcstGFSO_obsBDECKS_GDF_TDF/ (relative to **OUTPUT_BASE**)

For each month and year there will be five files written:

```
* tc_gen_2016_pairs.nc
* tc_gen_2016_genmpr.txt
* tc_gen_2016_ctc.txt
```

(continues on next page)

(continued from previous page)

```
* tc_gen_2016_cts.txt
* tc_gen_2016.stat
```

Keywords

Note:

- TCGenToolUseCase
- S2SAppUseCase
- UserScriptUseCase
- METplotpyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s-TCGen_fcstGFSO_obsBDECKS_GDF_TDF.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.10.4 UserScript: Make zonal and meridonial means

model_applications/ s2s/ UserScript_obsERA_obsOnly_Stratosphere.py

Scientific Objective

This use case calls functions in METcalcpy to create zonal and meridonial means

Datasets

SSWC_v1.0_varFull_ERAi_d20130106_s20121107_e20130307_c20160701.nc

METplus Components

This use case runs the UserScript wrapper tool to run a user provided script, in this case, meridonial.py.

METplus Workflow

This use case does not loop but plots the entire time period of data

UserScript This uses data from 20130106,20121107,20130307,20160701

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/s2s/UserScript_obsERA_obsOnly_Stratosphere.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s/UserScript_
↪obsERA_obsOnly_Stratosphere.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = REALTIME

VALID_TIME_FMT = %Y
VALID_BEG = 2013

```

(continues on next page)

(continued from previous page)

```
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/model_applications/s2s/UserScript_obsERA_obsOnly_
↪Stratosphere/meridonial_mean.py

[user_env_vars]
INPUT_FILE_NAME = {INPUT_BASE}/model_applications/s2s/UserScript_obsERA_obsOnly_Stratosphere/
↪SSWC_v1.0_varFull_ERAi_d20130106_s20121107_e20130307_c20160701.nc
YAML_CONFIG_NAME = {METPLUS_BASE}/parm/use_cases/model_applications/s2s/UserScript_obsERA_
↪obsOnly_Stratosphere/meridonial_mean.yaml

LOG_FILE = "Meridonial_means.log"

LOG_LEVEL = "INFO"

OUTPUT_DIR = {OUTPUT_BASE}
```

MET Configuration

There are no MET tools used in this use case.

Python Embedding

There is no python embedding in this use case

Running METplus

This use case can be run two ways:

1) Passing in meridonial_means.conf, then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s/UserScript_obsERA_
↪obsOnly_Stratosphere.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in meridonial.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s/UserScript_
↪obsERA_obsOnly_Stratosphere.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

  and for the [exe] section, you will need to define the location of NON-MET executables. If the executable is in the user's path, METplus will find it from the name. If the executable is not in the path, specify the full path to the executable here (i.e. RM = /bin/rm) The following executables are required for performing series analysis use cases:

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y

[exe]
RM = /path/to/rm
CUT = /path/to/cut
TR = /path/to/tr
NCAP2 = /path/to/ncap2
CONVERT = /path/to/convert
NCDUMP = /path/to/ncdump
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Keywords

Note:

- UserScriptUseCase
- S2SAppUseCase
- METcalcpyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/s2s-zonal_means.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.10.5 UserScript: Make a Hovmoeller plot

model_applications/ s2s/ UserScript_obsPrecip_obsOnly_Hovmoeller.py

Scientific Objective

This use case calls the METplotpy hovmoeller plot to create a sample Hovmoeller diagram using sample data created by METcalcpy hovmoeller functions

The Hovmoeller plot and hovmoeller calculations where created by Maria Gehne at the Physical Sciences Labratory in NOAA

Datasets

METplus Components

This use case runs the UserScript wrapper tool to run a user provided script, in this case, hovmoeller.py.

It also requires the METcalcpy and METplotpy source code to generate the plot. Clone the METcalcpy repository (https://github.com/dtcenter/METcalcpy) and the METplotpy repository (https://github.com/dtcenter/METplotpy) under the same base directory as the METPLUS_BASE directory so that the METplotpy, METcalcpy, and METplotpy directories are under the same base directory (i.e. if the METPLUS_BASE directory is /home/username/working/METplus, then clone the METcalcpy and METplotpy source code into the /home/username/working directory).

METplus Workflow

This use case does not loop but plots the entire time period of data

This uses data from 2016-01-01 to 2016-03-31

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line parm/use_cases/model_applications/s2s/UserScript_obsPrecip_obsOnly_Hovmoeller.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s/UserScript_
↪obsPrecip_obsOnly_Hovmoeller.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = REALTIME
VALID_TIME_FMT = %Y
VALID_BEG = 2014

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

```

(continues on next page)

(continued from previous page)

```
USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/model_applications/s2s/UserScript_obsPrecip_
↪obsOnly_Hovmoeller/hovmoeller_diagram.py


[user_env_vars]

# Difficulty index specific variables

LOG_FILE = "Hovmoeller_diagram.log"

LOG_LEVEL = "INFO"

YAML_CONFIG_NAME = {PARM_BASE}/use_cases/model_applications/s2s/UserScript_obsPrecip_obsOnly_
↪Hovmoeller/hovmoeller.yaml

INPUT_FILE_NAME = {INPUT_BASE}/model_applications/s2s/UserScript_obsPrecip_obsOnly_
↪Hovmoeller/precip.erai.sfc.1p0.2x.2014-2016.nc

OUTPUT_DIR = {OUTPUT_BASE}/plots
```

MET Configuration

There are no MET tools used in this use case.

Python Embedding

There is no python embedding in this use case

Running METplus

This use case can be run two ways:

1) Passing in UserScript_obsPrecip_obsOnly_Hovmoeller.conf, then a user-specific system configuration file:

```
run_metplus.py \
/path/to/METplus/parm/use_cases/model_applications/s2s/UserScript_obsPrecip_obsOnly_
↪Hovmoeller.conf \
/path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_obsPrecip_obsOnly_Hovmoeller.conf:

```
run_metplus.py \
/path/to/METplus/parm/use_cases/model_applications/s2s/UserScript_obsPrecip_obsOnly_
↪Hovmoeller.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

  and for the [exe] section, you will need to define the location of NON-MET executables. If the executable is in the user's path, METplus will find it from the name. If the executable is not in the path, specify the full path to the executable here (i.e. RM = /bin/rm) The following executables are required for performing series analysis use cases:

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y

[exe]
RM = /path/to/rm
CUT = /path/to/cut
TR = /path/to/tr
NCAP2 = /path/to/ncap2
CONVERT = /path/to/convert
NCDUMP = /path/to/ncdump
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Keywords

Note:

- UserScriptUseCase
- S2SAppUseCase
- METplotpyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/Hovmoeller_ERAIprecip_2016-01-01-2016-03-31.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.10.6 GridStat: Determine dominant ensemble members terciles and calculate categorical outputs

model_applications/s2s/GridStat_fcstCFSv2_obsGHCNCAMS_MultiTercile.conf

Scientific Objective

This use case ingests a CFSv2 Ensemble forecast, with all ensemble members in a single file for a given year. 29 years of forecast ensembles are used to create probabilities for each tercile, which is accomplished by a Python script. Of the terciles, each gridpoint is assigned a value corresponding to the tercile that is most likely to occur. This is compared to an observation set that contains the tercile data and MCTS line type is requested. This use case highlights the inclusion of tercile data for calculating HSS; in particular, how to utilize the hss_ec_value option to preset the expected values rather than relying on categorical values.

Datasets

Forecast: 29 CFSv2 Ensemble files, 2m temperature fields

Observations: GHCNCAMS, 2m temperature field

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1582) section for more information.

Data Source: CPC

METplus Components

This use case calls a Python script 29 times, once for each year of data of the CFSv2 ensemble. Each time a successful call to the script is made, a grid of 1s, 2s, and 3s is returned, representing which tercile was dominant for the gridpoint. GridStat processes the forecast and observation fields, and outputs the requested line types.

METplus Workflow

This use case utilizes 29 years of forecast data, with 24 members in each ensemble forecast. The following boundary times are used for the entire script:

Init Beg: 1982-01-01
Init End: 2010-01-02

Because the increment is 1 year, all January 1st from 1982 to 2010 are processed for a total of 29 years.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. -c parm/use_cases/model_applications/s2s/GridStat_fcstCFSv2_obsGHCNCAMS_MultiTercile.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s/GridStat_
↪fcstCFSv2_obsGHCNCAMS_MultiTercile.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###
```

(continues on next page)

(continued from previous page)

```
PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=1982010100
INIT_END=2010020100
INIT_INCREMENT = 1Y

LEAD_SEQ =


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_CLIMO_MEAN_INPUT_DIR =
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/HSS_out_Mplus
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m}
```

(continues on next page)

(continued from previous page)

```
###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = CFSv2
OBTYPE = OBS

FCST_VAR1_NAME = {CONFIG_DIR}/forecast_read-in_CFSv2_categoricalthresholds.py {INPUT_BASE}/
↪model_applications/s2s/GridStat_fcstCFSv2_obsGHCNCAMS_MultiTercile/CFSv2.tmp2m.{init?fmt=%Y
↪%m}.fcst.nc:tmp2m:{init?fmt=%Y%m%d%H}:0:0
FCST_VAR1_LEVELS =
FCST_VAR1_THRESH = lt1.5, lt2.5

OBS_VAR1_NAME = {CONFIG_DIR}/forecast_read-in_CFSv2_categoricalthresholds_obs.py {INPUT_BASE}
↪/model_applications/s2s/GridStat_fcstCFSv2_obsGHCNCAMS_MultiTercile/CFSv2.tmp2m.{init?fmt=
↪%Y%m}.fcst.nc:tmp2m:{init?fmt=%Y%m%d%H}:0:0
OBS_VAR1_LEVELS =
OBS_VAR1_THRESH = lt1.5, lt2.5

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/s2s/GridStat_fcstCFSv2_obsGHCNCAMS_
↪MultiTercile


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_REGRID_TO_GRID = FCST


GRID_STAT_DESC = NA

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_ONCE_PER_FIELD = False

GRID_STAT_OUTPUT_PREFIX =

GRID_STAT_OUTPUT_FLAG_MCTC = BOTH
GRID_STAT_OUTPUT_FLAG_MCTS = BOTH
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_NC_PAIRS_FLAG_LATLON = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
GRID_STAT_NC_PAIRS_FLAG_DIFF = TRUE

GRID_STAT_HSS_EC_VALUE =
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
```

(continues on next page)

(continued from previous page)

```
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstCFSv2_obsGHCNCAMS_MultiTercile.conf then a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/s2s/GridStat_
↪fcstCFSv2_obsGHCNCAMS_MultiTercile /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstCFSv2_obsGHCNCAMS_MultiTercile:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/s2s/GridStat_
↪fcstCFSv2_obsGHCNCAMS_MultiTercile.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[config]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for the use case will be found in 29 folders(relative to **OUTPUT_BASE**). The output will follow the time information of the run. Specifically:

- YYYY01

where YYYY will be replaced by values corresponding to each of the years (1982 through 2010). Each of those folders will have the following files:

- grid_stat_000000L_19820101_000000V_pairs.nc
- grid_stat_000000L_19820101_000000V_mctc.txt
- grid_stat_000000L_19820101_000000V_mcts.txt
- grid_stat_000000L_19820101_000000V.stat

Keywords

Note:

- GridStatToolUseCase
- ProbabilityVerificationUseCase
- PythonEmbeddingFileUseCase
- S2SAppUseCase
- NETCDFFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/s2s-GridStat_fcstCFSv2_obsGHCNCAMS_MultiTercile.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.10.7 SeriesAnalysis: Standardize ensemble members and calculate probabilistic outputs

model_applications/s2s/SeriesAnalysis_fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool.conf

Scientific Objective

This use case ingests a CFSv2 Ensemble forecast, with all ensemble members in a single file for a given year. 29 years of forecast ensembles are used to create climatologies for each ensemble member. These climatologies are then used to normalize each ensemble member via the Gen-Ens-Prod tool, allowing a meaningful comparison to the observation dataset, which is presented as normalized. The forecast to observation verification are completed across both the temporal and spatial. This use case highlights several important features within METplus; in particular, how to create climatologies for ensemble members using SeriesAnalysis, how those climatologies can be used by GenEnsProd to normalize each ensemble member to its corresponding climatology, and calculating probabilistic verfication on s2s data, which is a frequent request from climatological centers.

Datasets

Forecast: 29 CFSv2 Ensemble files, 2m temperature fields

Observations: GHCNCAMS, 2m temperature field

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1601) section for more information.

Data Source: CPC

METplus Components

This use case initially runs SeriesAnalysis 24 times, once for each member of the CFSv2 ensemble, across the entire 29 years for forecast data. The resulting 24 outputs are read in by GenEnsProd, which is called 29 times (once for each year). GenEnsProd uses the **normalize** option and the SeriesAnalysis outputs to normalize each of the ensemble members relative to its climatology (FBAR) and standard deviation (FSTDEV). The output from GenEnsProd are 29 files containing the uncalibrated probability forecasts for the lower tercile of January for each year. The final probability verification is done across the temporal scale in SeriesAnalysis, and the spatial scale in GridStat.

METplus Workflow

This use case utilizes 29 years of forecast data, with 24 members in each ensemble forecast. The following boundary times are used for the entire script:

Init Beg: 1982-01-01
Init End: 2010-01-02

Because the increment is 1 year, all January 1st from 1982 to 2010 are processed for a total of 29 years.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. -c parm/use_cases/model_applications/s2s/SeriesAnalysis_fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatistic

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s/SeriesAnalysis_
↪fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
```

(continues on next page)

(continued from previous page)

```
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = SeriesAnalysis, GenEnsProd, SeriesAnalysis(run_two), GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m
INIT_BEG=198201
INIT_END=201002
INIT_INCREMENT = 1Y

LEAD_SEQ =

SERIES_ANALYSIS_CUSTOM_LOOP_LIST = 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,
↪22,23
SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

# SeriesAnalysis

FCST_SERIES_ANALYSIS_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/SeriesAnalysis_
↪fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool
FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = CFSv2.tmp2m.{init?fmt=%Y%m}.fcst.nc

OBS_SERIES_ANALYSIS_INPUT_DIR = {FCST_SERIES_ANALYSIS_INPUT_DIR}
```

(continues on next page)

(continued from previous page)

```
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = {FCST_SERIES_ANALYSIS_INPUT_TEMPLATE}

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/SA_run1
SERIES_ANALYSIS_OUTPUT_TEMPLATE = mem{custom?fmt=%s}_output.nc


# GenEnsProd

GEN_ENS_PROD_INPUT_DIR = {FCST_SERIES_ANALYSIS_INPUT_DIR}
GEN_ENS_PROD_INPUT_TEMPLATE = {FCST_SERIES_ANALYSIS_INPUT_TEMPLATE}

GEN_ENS_PROD_N_MEMBERS = 24

GEN_ENS_PROD_CLIMO_MEAN_FILE_NAME = {SERIES_ANALYSIS_OUTPUT_DIR}/memMET_ENS_MEMBER_ID_output.
↪nc
GEN_ENS_PROD_CLIMO_MEAN_FIELD = {name="series_cnt_FBAR"; level="(*,*)";}

GEN_ENS_PROD_CLIMO_STDEV_FILE_NAME = {SERIES_ANALYSIS_OUTPUT_DIR}/memMET_ENS_MEMBER_ID_
↪output.nc
GEN_ENS_PROD_CLIMO_STDEV_FIELD = {name="series_cnt_FSTDEV"; level="(*,*)";}

GEN_ENS_PROD_OUTPUT_DIR = {OUTPUT_BASE}/GEP
GEN_ENS_PROD_OUTPUT_TEMPLATE = gen_ens_prod_{init?fmt=%Y%m}_ens.nc


# SeriesAnalysis(run_two)

[run_two]

FCST_SERIES_ANALYSIS_INPUT_DIR = {OUTPUT_BASE}/GEP
FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = gen_ens_prod_{init?fmt=%Y%m}_ens.nc

OBS_SERIES_ANALYSIS_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/SeriesAnalysis_fcstCFSv2_
↪obsGHCNCAMS_climoStandardized_MultiStatisticTool
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = ghcn_cams.1x1.1982-2020.mon.nc

SERIES_ANALYSIS_CLIMO_MEAN_FILE_NAME = {INPUT_BASE}/model_applications/s2s/SeriesAnalysis_
↪fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool/ghcn_cams.1x1.1982-2010.mon.
↪clim.nc
SERIES_ANALYSIS_CLIMO_MEAN_FIELD = {name="clim"; level="(0,*,*)";}
SERIES_ANALYSIS_CLIMO_MEAN_FILE_TYPE = NETCDF_NCCF

SERIES_ANALYSIS_CLIMO_STDEV_FILE_NAME = {INPUT_BASE}/model_applications/s2s/SeriesAnalysis_
↪fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool/ghcn_cams.1x1.1982-2010.mon.
↪stddev.nc
```

(continues on next page)

(continued from previous page)

```
SERIES_ANALYSIS_CLIMO_STDEV_FIELD = {name="stddev"; level="(0,*,*)";}
SERIES_ANALYSIS_CLIMO_STDEV_FILE_TYPE = NETCDF_NCCF

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/SA_run2
SERIES_ANALYSIS_OUTPUT_TEMPLATE = {INIT_BEG}to{INIT_END}_CFSv2_SA.nc


# GridStat

[config]

FCST_GRID_STAT_INPUT_DIR = {GEN_ENS_PROD_OUTPUT_DIR}
FCST_GRID_STAT_INPUT_TEMPLATE = {GEN_ENS_PROD_OUTPUT_TEMPLATE}

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/SeriesAnalysis_fcstCFSv2_
↪obsGHCNCAMS_climoStandardized_MultiStatisticTool
OBS_GRID_STAT_INPUT_TEMPLATE = ghcn_cams.1x1.1982-2020.mon.nc

GRID_STAT_CLIMO_MEAN_FILE_NAME = {INPUT_BASE}/model_applications/s2s/SeriesAnalysis_
↪fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool/ghcn_cams.1x1.1982-2010.mon.
↪clim.nc
GRID_STAT_CLIMO_MEAN_FIELD = {name="clim"; level="(0,*,*)";}
GRID_STAT_CLIMO_MEAN_FILE_TYPE = NETCDF_NCCF

GRID_STAT_CLIMO_STDEV_FILE_NAME = {INPUT_BASE}/model_applications/s2s/SeriesAnalysis_
↪fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool/ghcn_cams.1x1.1982-2010.mon.
↪stddev.nc
GRID_STAT_CLIMO_STDEV_FIELD = {name="stddev"; level="(0,*,*)";}
GRID_STAT_CLIMO_STDEV_FILE_TYPE = NETCDF_NCCF

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = CFSv2


# SeriesAnalysis

BOTH_SERIES_ANALYSIS_VAR1_NAME = fcst
```

(continues on next page)

(continued from previous page)

```
BOTH_SERIES_ANALYSIS_VAR1_LEVELS = "({custom},0,*,*)"
SERIES_ANALYSIS_FCST_FILE_TYPE = NETCDF_NCCF
SERIES_ANALYSIS_OBS_FILE_TYPE = NETCDF_NCCF


# GenEnsProd

ENS_VAR1_NAME = fcst
ENS_VAR1_LEVELS = "(MET_ENS_MEMBER_ID,0,*,*)"
ENS_VAR1_THRESH = <-0.43, >=-0.43&&<=0.43, >0.43
ENS_FILE_TYPE = NETCDF_NCCF


# SeriesAnalysis(run_two)

[run_two]

FCST_SERIES_ANALYSIS_VAR1_NAME = fcst_0_0_all_all_ENS_FREQ_lt-0.43
FCST_SERIES_ANALYSIS_VAR1_LEVELS = "(*,*)"

FCST_CAT_THRESH = ==0.1
FCST_IS_PROB = True

OBS_SERIES_ANALYSIS_VAR1_NAME = tmp2m
OBS_SERIES_ANALYSIS_VAR1_LEVELS = "({init?fmt=%Y%m%d_%H%M%S},*,*)"
OBS_SERIES_ANALYSIS_CAT_THRESH = <=CDP33

OBS_FILE_TYPE = NETCDF_NCCF


# GridStat

[config]
FCST_GRID_STAT_VAR1_NAME = fcst_0_0_all_all_ENS_FREQ_lt-0.43
FCST_GRID_STAT_VAR1_LEVELS = "(*,*)"
FCST_GRID_STAT_VAR1_THRESH = ==0.1
FCST_GRID_STAT_IS_PROB = True

OBS_GRID_STAT_VAR1_NAME = tmp2m
OBS_GRID_STAT_VAR1_LEVELS = "({init?fmt=%Y%m%d_%H%M%S},*,*)"
OBS_GRID_STAT_VAR1_THRESH = <=CDP33
OBS_GRID_STAT_FILE_TYPE = NETCDF_NCCF


###
```

(continues on next page)

(continued from previous page)

```
# SeriesAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#seriesanalysis
###

SERIES_ANALYSIS_OUTPUT_STATS_CNT = TOTAL, FBAR, FSTDEV
SERIES_ANALYSIS_BLOCK_SIZE = 0


###
# GenEnsProd Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#genensprod
###

GEN_ENS_PROD_NORMALIZE = CLIMO_STD_ANOM

GEN_ENS_PROD_ENS_THRESH = 0.3
GEN_ENS_PROD_VLD_THRESH = 0.3

GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY = TRUE

GEN_ENS_PROD_ENS_MEMBER_IDS = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,␣
↪18, 19, 20, 21, 22, 23


###
# SeriesAnalysis(run_two) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#seriesanalysis
###

[run_two]

SERIES_ANALYSIS_REGRID_TO_GRID = FCST
SERIES_ANALYSIS_OUTPUT_STATS_PSTD = TOTAL, BRIER, RELIABILITY, BRIERCL, BSS
SERIES_ANALYSIS_VLD_THRESH = 0.5

SERIES_ANALYSIS_BLOCK_SIZE = 0

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False
```

(continues on next page)

(continued from previous page)

```
###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

[config]

GRID_STAT_OUTPUT_PREFIX = {init?fmt=%Y%m}

GRID_STAT_REGRID_TO_GRID = FCST
GRID_STAT_OUTPUT_FLAG_PSTD = BOTH
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = TRUE
GRID_STAT_NC_PAIRS_FLAG_RAW = TRUE
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

```
////////////////////////////////////////////////////////////////////////////////
//
// Series-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
//
//desc =
${METPLUS_DESC}
```

(continues on next page)

(continued from previous page)

```
//
// Output observation type to be written
//
//obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

censor_thresh = [];
censor_val    = [];
//cat_thresh =
${METPLUS_CAT_THRESH}
cnt_thresh    = [ NA ];
cnt_logic     = UNION;

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_CAT_THRESH}
   ${METPLUS_FCST_FIELD}
}
obs = {
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}
```

(continues on next page)

(continued from previous page)

```
//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

//
// Number of grid points to be processed concurrently.  Set smaller to use
// less memory but increase the number of passes through the data.
//
//block_size =
${METPLUS_BLOCK_SIZE}

//
// Ratio of valid matched pairs to compute statistics for a grid point
//
//vld_thresh =
${METPLUS_VLD_THRESH}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Statistical output types
//
//output_stats = {
${METPLUS_OUTPUT_STATS_DICT}

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

```
////////////////////////////////////////////////////////////////////////////////
//
// Gen-Ens-Prod configuration file.
//
// For additional information, please see the MET Users Guide.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
//desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
//censor_thresh =
${METPLUS_CENSOR_THRESH}

//censor_val    =
${METPLUS_CENSOR_VAL}

//normalize =
${METPLUS_NORMALIZE}

//cat_thresh    =
${METPLUS_CAT_THRESH}

//nc_var_str    =
${METPLUS_NC_VAR_STR}

//
// Ensemble fields to be processed
//
ens = {
   //file_type =
   ${METPLUS_ENS_FILE_TYPE}

   //ens_thresh =
   ${METPLUS_ENS_THRESH}

   //vld_thresh =
   ${METPLUS_VLD_THRESH}

   //field =
   ${METPLUS_ENS_FIELD}

}

//ens_member_ids =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}


////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood ensemble probabilities
//
//nbrhd_prob = {
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
//nmep_smooth = {
${METPLUS_NMEP_SMOOTH_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}

//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Ensemble product output types
// May be set separately in each "ens.field" entry
//
//ensemble_flag = {
${METPLUS_ENSEMBLE_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//version = "V10.1.0";

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
```

(continues on next page)

(continued from previous page)

```
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in SeriesAnalysis_fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool.conf then a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/s2s/SeriesAnalysis_
↪fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool.conf /path/to/user_system.
↪conf
```

2) Modifying the configurations in parm/metplus_config, then passing in SeriesAnalysis_fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool.conf:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/marine_and_cryosphere/
↪SeriesAnalysis_fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[config]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for use case will be found in 4 distinct folders (relative to **OUTPUT_BASE**). The output from the first SeriesAnalysis call goes to **SA_run1** will contain the following files:

- mem??_output.nc

where ?? will be replaced by values corresponding to each of the ensemble members (0 through 23). The output for GenEnsProd goes into **GEP** and contains the following files:

- gen_ens_prod_YYYY01_ens.nc

where YYYY will be replaced by each year of the forecast data being processed (1982 through 2010). The output from the second SeriesAnalysis call goes to **SA_run2** and contains the following files:

- 198201to201002_CFSv2_SA.nc

Finally, the output from GridStat will be in **GridStat** and will contain 29 folders of the following format:

- ????01

where ???? will correspond to each year of the forecast data being processed (1982 through 2010). Each of those folders will have the following files:

- grid_stat_198201_000000L_19700101_000000V_pairs.nc
- grid_stat_198201_000000L_19700101_000000V_pstd.txt
- grid_stat_198201_000000L_19700101_000000V.stat

Keywords

Note:

- SeriesAnalysisUseCase
- GenEnsProdUseCase
- GridStatUseCase
- ProbabilityVerificationUseCase
- S2SAppUseCase
- NETCDFFileUseCase
- ClimatologyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s-SeriesAnalysis_fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiSta

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.11 Subseasonal to Seasonal: Mid-Latitude

Subseasonal-to-Seasonal model configurations relating to middle latitudes

7.2.16.11.1 Blocking Calculation: ERA RegridDataPlane, PcpCombine, and Blocking python code

model_applications/ s2s_mid_lat/ UserScript_obsERA_obsOnly_Blocking.py

Scientific Objective

To compute the frequency of blocking using the Pelly-Hoskins method. Specifically the blocking calculation consits of computing the Central Blocking Latitude (CBL), Instantaneousy blocked latitudes (IBL), Group Instantaneousy blocked latitudes (GIBL), and the frequency of atmospheric blocking. The CBL calculation had an option to use an observed climatology.

The following reference contains the specific equations and methodology used to compute blocking:

Datasets

- Forecast dataset: None
- Observation dataset: ERA Reanlaysis 500 mb height.

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* numpy
* netCDF4
* datetime
* bisect
* scipy
```

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case runs the blocking driver script which runs the steps the user lists in STEPS_OBS. The possible steps are regridding, time averaging, computing a running mean, computing anomalies, computing CBLs (CBL), plotting CBLs (PLOTCBL), computing IBLs (IBL), plotting IBL frequency (PLOTIBL), computing GIBLs (GIBL), computing blocks (CALCBLOCKS), and plotting the blocking frequency (PLOTBLOCKS). Regridding, time averaging, running means, and anomaloies are set up in the UserScript .conf file and are formatted as follows: PROCESS_LIST = RegridDataPlane(regrid_obs), PcpCombine(daily_mean_obs), PcpCombine(running_mean_obs), PcpCombine(anomaly_obs), UserScript(script_blocking)

The other steps are listed in the Blocking .conf file and are formatted as follows: OBS_STEPS = CBL+PLOTCBL+IBL+PLOTIBL+GIBL+CALCBLOCKS+PLOTBLOCKS

METplus Workflow

The blocking python code is run for each time for the forecast and observations data. This example loops by valid time. This version is set to only process the blocking steps (CBL, PLOTCBL, IBL, PLOTIBL, GIBL, CALCBLOCKS, PLOTBLOCKS), omitting the regridding, time averaging, running mean, and anomaly pre-processing steps. However, the configurations for pre-processing are available for user reference.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/s2s_mid_lat/UserScript_obsERA_obsOnly_Blocking.py. The file UserScript_obsERA_obsOnly_Blocking.conf runs the python program, and the variables for all steps of the Blocking use case are set in the [user_env_vars] section.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s_mid_lat/
↪UserScript_obsERA_obsOnly_Blocking.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

# All steps, including pre-processing:
# PROCESS_LIST = RegridDataPlane(regrid_obs), PcpCombine(daily_mean_obs), PcpCombine(running_
↪mean_obs), PcpCombine(anomaly_obs), UserScript(script_blocking)
# Only Blocking Analysis script for the observations

PROCESS_LIST = UserScript(script_blocking)


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 1979120100
VALID_END = 2017022800
VALID_INCREMENT = 86400
```

(continues on next page)

(continued from previous page)

```
LEAD_SEQ = 0

# Only Process DJF
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:0229"

# Run the obs data
# A variable set to be used in the pre-processing steps
OBS_RUN = True


###
# RegridDataPlane(regrid_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Regrid the observations to 1 degree using regrid_data_plane
[regrid_obs]

VALID_END = 2017022818
VALID_INCREMENT = 21600

OBS_REGRID_DATA_PLANE_RUN = {OBS_RUN}

OBS_DATA_PLANE_ONCE_PER_FIELD = False

OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Z
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = P500
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = Z500

REGRID_DATA_PLANE_VERIF_GRID = latlon 360 90 89 0 -1.0 1.0

REGRID_DATA_PLANE_METHOD = BILIN

REGRID_DATA_PLANE_WIDTH = 2

OBS_REGRID_DATA_PLANE_INPUT_DIR = /gpfs/fs1/collections/rda/data/ds627.0/ei.oper.an.pl
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪Blocking/ERA/Regrid

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = {valid?fmt=%Y%m}/ei.oper.an.pl.regn128sc.{valid?fmt=%Y
↪%m%d%H}
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/Z500_6hourly_{init?fmt=%Y%m%d%H}_
↪NH.nc
```

(continues on next page)

(continued from previous page)

```
###
# PCPCombine(daily_mean_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Perform a sum over the 4 daily times that have been regridded using pcp_combine
# 00, 06, 12, 18 UTC
[daily_mean_obs]

VALID_BEG = 1979120118
VALID_END = 2017022818

OBS_PCP_COMBINE_RUN = {OBS_RUN}

OBS_PCP_COMBINE_METHOD = DERIVE
OBS_PCP_COMBINE_STAT_LIST = MEAN

OBS_PCP_COMBINE_INPUT_ACCUMS = 6
OBS_PCP_COMBINE_INPUT_NAMES = Z500
OBS_PCP_COMBINE_INPUT_LEVELS = "(*,*)"
OBS_PCP_COMBINE_INPUT_OPTIONS = convert(x) = x / 9.81; set_attr_valid = "{valid?fmt=%Y%m%d_%H
↪%M%S?shift=-64800}";

OBS_PCP_COMBINE_OUTPUT_ACCUM = 24
OBS_PCP_COMBINE_DERIVE_LOOKBACK = 24

OBS_PCP_COMBINE_OUTPUT_NAME = Z500

OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/ERA/
↪Regrid
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/
↪ERA/Daily

OBS_PCP_COMBINE_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/Z500_6hourly_{valid?fmt=%Y%m%d%H}_NH.nc
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = Z500_daily_{valid?fmt=%Y%m%d?shift=-64800}_NH.nc


###
# PCPCombine(running_mean_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Perform a 5 day running mean on the data using pcp_combine
```

(continues on next page)

(continued from previous page)

```
[running_mean_obs]

# Add the first/last 2 days to the skip times to compute the running mean
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:1201,1202,1203,1204,0229"

OBS_PCP_COMBINE_RUN = {OBS_RUN}

OBS_PCP_COMBINE_METHOD = DERIVE
OBS_PCP_COMBINE_STAT_LIST = MEAN

OBS_PCP_COMBINE_INPUT_ACCUMS = 24
OBS_PCP_COMBINE_INPUT_NAMES = Z500
OBS_PCP_COMBINE_INPUT_LEVELS = "(*,*)"
OBS_PCP_COMBINE_INPUT_OPTIONS = set_attr_valid = "{valid?fmt=%Y%m%d_%H%M%S?shift=-172800}";

#  Running mean is 5 days
OBS_PCP_COMBINE_OUTPUT_ACCUM = 120
OBS_PCP_COMBINE_DERIVE_LOOKBACK = 120

OBS_PCP_COMBINE_OUTPUT_NAME = Z500

OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/ERA/
↪Daily
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/
↪ERA/Rmean5d

OBS_PCP_COMBINE_INPUT_TEMPLATE = Z500_daily_{valid?fmt=%Y%m%d}_NH.nc
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = Z500_5daymean_{valid?fmt=%Y%m%d?shift=-172800}_NH.nc


###
# PCPCombine(anomaly_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Compute anomalies using the daily means and 5 day running mean using pcp_combine

[anomaly_obs]

# Add the first/last 2 days to the skip times to compute the running mean
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:1201,1202,0227,0228,0229"

OBS_PCP_COMBINE_RUN = {OBS_RUN}

OBS_PCP_COMBINE_METHOD = USER_DEFINED
```

(continues on next page)

(continued from previous page)

```
OBS_PCP_COMBINE_COMMAND = -subtract {OBS_PCP_COMBINE_INPUT_DIR}/Daily/Z500_daily_{valid?fmt=
↪%Y%m%d}_NH.nc {OBS_PCP_COMBINE_INPUT_DIR}/Rmean5d/Z500_5daymean_{valid?fmt=%Y%m%d}_NH.nc -
↪field 'name="Z500"; level="(*,*)";'

OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/ERA
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/
↪ERA/Anomaly

OBS_PCP_COMBINE_INPUT_TEMPLATE = Z500_daily_{valid?fmt=%Y%m%d}_NH.nc
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = Z500_anomaly_{valid?fmt=%Y%m%d}_NH.nc


# Variables set for the Blocking Analysis
[user_env_vars]
# Steps to Run
OBS_STEPS = CBL+PLOTCBL+IBL+PLOTIBL+GIBL+CALCBLOCKS+PLOTBLOCKS

# Number of Seasons and Days per season that should be available
# The code will fill missing data, but requires the same number of days per
# season for each year.  You may need to omit leap days if February is part of
# the processing
CBL_NUM_SEASONS = 38
IBL_NUM_SEASONS = 38
DAYS_PER_SEASON = 86

# Make the OUTPUT_BASE available to the UserScript
SCRIPT_OUTPUT_BASE = {OUTPUT_BASE}

# Variable Name for the Z500 anomaly data to read in to the blocking python code
OBS_BLOCKING_ANOMALY_VAR = Z500_ANA

# Variable for the Z500 data
OBS_BLOCKING_VAR = Z500

# Number of model grid points used for a moving average
# Must be odd
OBS_SMOOTHING_PTS = 9

# Lat Delta, to allow for offset from the Central Blocking Latitude
OBS_LAT_DELTA = -5,0,5

# Meridional Extent of blocks (NORTH_SOUTH_LIMITS/2)
OBS_NORTH_SOUTH_LIMITS = 30
```

(continues on next page)

(continued from previous page)

```
# Maximum number of grid points between IBLs for everything in between to be included as an␣
↪IBL
OBS_IBL_DIST = 7

# Number of grid points in and IBL to make a GIBL
OBS_IBL_IN_GIBL = 15

# Number of grid points that must overlap across days for a GIBL
OBS_GIBL_OVERLAP = 10

# Time duration in days needed for a block
OBS_BLOCK_TIME = 5

# Number of grid points a block must travel to terminate
OBS_BLOCK_TRAVEL = 45

# Method to compute blocking.  Currently, the only option is 'PH' for the
# Pelly-Hoskins Method
OBS_BLOCK_METHOD = PH

# Plot Output Directory
BLOCKING_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_obsERA_obsOnly_Blocking/
↪plots/

#CBL plot title and name
OBS_CBL_PLOT_MTHSTR = DJF
OBS_CBL_PLOT_OUTPUT_NAME = ERA_CBL_avg

# IBL plot title and name
OBS_IBL_PLOT_TITLE = DJF ERA Instantaneous Blocked Longitude
OBS_IBL_PLOT_OUTPUT_NAME = ERA_IBL_Freq_DJF

# Blocking plot title and name
OBS_BLOCKING_PLOT_TITLE = DJF ERA Blocking Frequency
OBS_BLOCKING_PLOT_OUTPUT_NAME = ERA_Block_Freq_DJF


###
# UserScript(script_blocking) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Run the Blocking Analysis Script
[script_blocking]
```

(continues on next page)

(continued from previous page)

```
# Skip the days on the edges that are not available due to the running mean
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:1201,1202,0227,0228,0229"

# Run the user script once per lead
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

# Template of filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mid_lat/UserScript_fcstGFS_
↪obsERA_Blocking/ERA/Anomaly/Z500_anomaly_{valid?fmt=%Y%m%d}_NH.nc,{INPUT_BASE}/model_
↪applications/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/ERA/Daily/Z500_daily_{valid?
↪fmt=%Y%m%d}_NH.nc

# Name of the file containing the listing of input files
# The options are OBS_CBL_INPUT, FCST_CBL_INPUT, OBS_IBL_INPUT, and FCST_IBL_INPUT
# *** Make sure the order is the same as the order of templates listed in USER_SCRIPT_INPUT_
↪TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS = OBS_CBL_INPUT,OBS_IBL_INPUT

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_obsERA_obsOnly_Blocking/Blocking_driver.py
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

See the following files for more information about the environment variables set in this configuration file.

parm/use_cases/met_tool_wrapper/RegridDataPlane/RegridDataPlane.py parm/use_cases/met_tool_wrapper/PCPCo
parm/use_cases/met_tool_wrapper/PCPCombine/PCPCombine_subtract.py

Python Scripts

This use case uses Python scripts to perform the blocking calculation

parm/use_cases/model_applications/s2s_mid_lat/UserScript_obsERA_obsOnly_Blocking/Blocking_driver.py: This script calls the requested steps in the blocking analysis for a forecast, observation, or both.

metcalcpy/contributed/blocking_weather_regime/Blocking.py: This script runs the requested steps, containing the code for computing CBLs, computing IBLs, computing GIBLs, and computing blocks. See the METcalcpy Blocking Calculation Script for more information.

metcalcpy/contributed/blocking_weather_regime/Blocking_WeatherRegime_util.py: This script contains functions used by both the blocking anwd weather regime analysis, including the code for determining which steps the user wants to run, and finding and reading the input files in the format from the output pre-processing steps. See the METcalcpy Utility script for more information.

```
#!/usr/bin/env python3
import sys
import os
import numpy as np
import datetime
import netCDF4
import warnings

from metcalcpy.contributed.blocking_weather_regime.Blocking import BlockingCalculation
from metcalcpy.contributed.blocking_weather_regime.Blocking_WeatherRegime_util import parse_
↪steps, write_mpr_file
from metplotpy.contributed.blocking_s2s import plot_blocking as pb
from metplotpy.contributed.blocking_s2s.CBL_plot import create_cbl_plot


def main():

    steps_list_fcst,steps_list_obs = parse_steps()

    if not steps_list_obs and not steps_list_fcst:
        warnings.warn('No processing steps requested for either the model or observations,')
        warnings.warn(' nothing will be run')
        warnings.warn('Set FCST_STEPS and/or OBS_STEPS in the [user_env_vars] section to␣
↪process data')


    ######################################################################
    # Blocking Calculation and Plotting
    ######################################################################
    # Set up the data
    steps_fcst = BlockingCalculation('FCST')
    steps_obs = BlockingCalculation('OBS')
```

(continues on next page)

(continued from previous page)

```
    # Check to see if there is a plot directory
    oplot_dir = os.environ.get('BLOCKING_PLOT_OUTPUT_DIR','')
    if not oplot_dir:
        obase = os.environ['SCRIPT_OUTPUT_BASE']
        oplot_dir = os.path.join(obase,'plots')
    if not os.path.exists(oplot_dir):
        os.makedirs(oplot_dir)

    # Check to see if there is a mpr output directory
    mpr_dir = os.environ.get('BLOCKING_MPR_OUTPUT_DIR','')
    if not mpr_dir:
        obase = os.environ['SCRIPT_OUTPUT_BASE']
        mpr_dir = os.path.join(obase,'mpr')

    # Check to see if CBL's are used from an obs climatology
    use_cbl_obs = os.environ.get('USE_CBL_OBS','False').lower()

    # Get the days per season
    dseasons = int(os.environ['DAYS_PER_SEASON'])

    # Grab the Anomaly (CBL) text files
    obs_cbl_filetxt = os.environ.get('METPLUS_FILELIST_OBS_CBL_INPUT','')
    fcst_cbl_filetxt = os.environ.get('METPLUS_FILELIST_FCST_CBL_INPUT','')

    # Grab the Daily (IBL) text files
    obs_ibl_filetxt = os.environ.get('METPLUS_FILELIST_OBS_IBL_INPUT','')
    fcst_ibl_filetxt = os.environ.get('METPLUS_FILELIST_FCST_IBL_INPUT','')


    # Calculate Central Blocking Latitude
    if ("CBL" in steps_list_obs):
        print('Computing Obs CBLs')
        # Read in the list of CBL files
        cbl_nseasons = int(os.environ['CBL_NUM_SEASONS'])
        with open(obs_cbl_filetxt) as ocl:
            obs_infiles = ocl.read().splitlines()
        if (obs_infiles[0] == 'file_list'):
            obs_infiles = obs_infiles[1:]
        if len(obs_infiles) != (cbl_nseasons*dseasons):
            raise Exception('Invalid Obs data; each year must contain the same date range to␣
↪calculate seasonal averages.')
        cbls_obs,lats_obs,lons_obs,mhweight_obs,cbl_time_obs = steps_obs.run_CBL(obs_infiles,
↪cbl_nseasons,dseasons)
```

(continues on next page)

(continued from previous page)

```
    if ("CBL" in steps_list_fcst) and (use_cbl_obs == 'false'):
        # Add in step to use obs for CBLS
        print('Computing Forecast CBLs')
        cbl_nseasons = int(os.environ['CBL_NUM_SEASONS'])
        with open(fcst_cbl_filetxt) as fcl:
            fcst_infiles = fcl.read().splitlines()
        if (fcst_infiles[0] == 'file_list'):
            fcst_infiles = fcst_infiles[1:]
        if len(fcst_infiles) != (cbl_nseasons*dseasons):
            raise Exception('Invalid Fcst data; each year must contain the same date range␣
↪to calculate seasonal averages.')
        cbls_fcst,lats_fcst,lons_fcst,mhweight_fcst,cbl_time_fcst = steps_fcst.run_CBL(fcst_
↪infiles,cbl_nseasons,dseasons)
    elif ("CBL" in steps_list_fcst) and (use_cbl_obs == 'true'):
        if not ("CBL" in steps_list_obs):
            raise Exception('Must run observed CBLs before using them as a forecast.')
        cbls_fcst = cbls_obs
        lats_fcst = lats_obs
        lons_fcst = lons_obs
        mhweight_fcst = mhweight_obs
        cbl_time_fcst = cbl_time_obs

    #Plot Central Blocking Latitude
    if ("PLOTCBL" in steps_list_obs):
        if not ("CBL" in steps_list_obs):
            raise Exception('Must run observed CBLs before plotting them.')
        print('Plotting Obs CBLs')
        cbl_plot_mthstr = os.environ['OBS_CBL_PLOT_MTHSTR']
        cbl_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_CBL_PLOT_OUTPUT_NAME',
↪'obs_cbl_avg'))
        create_cbl_plot(lons_obs, lats_obs, cbls_obs, mhweight_obs, cbl_plot_mthstr, cbl_
↪plot_outname,
            do_averaging=True)
    if ("PLOTCBL" in steps_list_fcst):
        if not ("CBL" in steps_list_fcst):
            raise Exception('Must run forecast CBLs before plotting them.')
        print('Plotting Forecast CBLs')
        cbl_plot_mthstr = os.environ['FCST_CBL_PLOT_MTHSTR']
        cbl_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_CBL_PLOT_OUTPUT_NAME',
↪'fcst_cbl_avg'))
        create_cbl_plot(lons_fcst, lats_fcst, cbls_fcst, mhweight_fcst, cbl_plot_mthstr, cbl_
↪plot_outname,
            do_averaging=True)
```

(continues on next page)

(continued from previous page)

```
    # Run IBL
    if ("IBL" in steps_list_obs):
        if not ("CBL" in steps_list_obs):
            raise Exception('Must run observed CBLs before running IBLs.')
        print('Computing Obs IBLs')
        ibl_nseasons = int(os.environ['IBL_NUM_SEASONS'])
        with open(obs_ibl_filetxt) as oil:
            obs_infiles = oil.read().splitlines()
        if (obs_infiles[0] == 'file_list'):
            obs_infiles = obs_infiles[1:]
        if len(obs_infiles) != (ibl_nseasons*dseasons):
            raise Exception('Invalid Obs data; each year must contain the same date range to␣
↪calculate seasonal averages.')
        ibls_obs,ibl_time_obs = steps_obs.run_Calc_IBL(cbls_obs,obs_infiles,ibl_nseasons,
↪dseasons)
        daynum_obs = np.arange(0,len(ibls_obs[0,:,0]),1)
    if ("IBL" in steps_list_fcst):
        if (not "CBL" in steps_list_fcst):
            raise Exception('Must run forecast CBLs or use observed CBLs before running IBLs.
↪')
        print('Computing Forecast IBLs')
        ibl_nseasons = int(os.environ['IBL_NUM_SEASONS'])
        with open(fcst_ibl_filetxt) as fil:
            fcst_infiles = fil.read().splitlines()
        if (fcst_infiles[0] == 'file_list'):
            fcst_infiles = fcst_infiles[1:]
        if len(fcst_infiles) != (ibl_nseasons*dseasons):
            raise Exception('Invalid Fcst data; each year must contain the same date range␣
↪to calculate seasonal averages.')
        ibls_fcst,ibl_time_fcst = steps_fcst.run_Calc_IBL(cbls_fcst,fcst_infiles,ibl_
↪nseasons,dseasons)
        daynum_fcst = np.arange(0,len(ibls_fcst[0,:,0]),1)

    if ("IBL" in steps_list_obs) and ("IBL" in steps_list_fcst):
        # Print IBLs to output matched pair file
        i_mpr_outdir = os.path.join(mpr_dir,'IBL')
        if not os.path.exists(i_mpr_outdir):
            os.makedirs(i_mpr_outdir)
        modname = os.environ.get('MODEL_NAME','GFS')
        maskname = os.environ.get('MASK_NAME','FULL')
        ibl_outfile_prefix = os.path.join(i_mpr_outdir,'IBL_stat_'+modname)
        cbls_avg = np.nanmean(cbls_obs,axis=0)
        write_mpr_file(ibls_obs,ibls_fcst,cbls_avg,lons_obs,ibl_time_obs,ibl_time_fcst,
↪modname,
            'NA','IBLs','block','Z500','IBLs','block','Z500',maskname,'500',ibl_outfile_
```

(continues on next page)

(continued from previous page)

```
↪prefix)

   # Plot IBLS
   if("PLOTIBL" in steps_list_obs) and not ("PLOTIBL" in steps_list_fcst):
       if not ("IBL" in steps_list_obs):
           raise Exception('Must run observed IBLs before plotting them.')
       print('Plotting Obs IBLs')
       ibl_plot_title = os.environ.get('OBS_IBL_PLOT_TITLE','Instantaneous Blocked Longitude
↪')
       ibl_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_IBL_PLOT_OUTPUT_NAME',
↪'obs_IBL_Freq'))
       ibl_plot_label1 = os.environ.get('IBL_PLOT_OBS_LABEL','')
       pb.plot_ibls(ibls_obs,lons_obs,ibl_plot_title,ibl_plot_outname,label1=ibl_plot_
↪label1)
   elif ("PLOTIBL" in steps_list_fcst) and not ("PLOTIBL" in steps_list_obs):
       if not ("IBL" in steps_list_fcst):
           raise Exception('Must run forecast IBLs before plotting them.')
       print('Plotting Forecast IBLs')
       ibl_plot_title = os.environ.get('FCST_IBL_PLOT_TITLE','Instantaneous Blocked␣
↪Longitude')
       ibl_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_IBL_PLOT_OUTPUT_NAME',
↪'fcst_IBL_Freq'))
       ibl_plot_label1 = os.environ.get('IBL_PLOT_FCST_LABEL','')
       pb.plot_ibls(ibls_fcst,lons_fcst,ibl_plot_title,ibl_plot_outname,label1=ibl_plot_
↪label1)
   elif ("PLOTIBL" in steps_list_obs) and ("PLOTIBL" in steps_list_fcst):
       if (not "IBL" in steps_list_obs) and (not "IBL" in steps_list_fcst):
           raise Exception('Must run forecast and observed IBLs before plotting them.')
       print('Plotting Obs and Forecast IBLs')
       ibl_plot_title = os.environ['IBL_PLOT_TITLE']
       ibl_plot_outname = os.path.join(oplot_dir,os.environ.get('IBL_PLOT_OUTPUT_NAME','IBL_
↪Freq'))
       #Check to see if there are plot legend labels
       ibl_plot_label1 = os.environ.get('IBL_PLOT_OBS_LABEL','Observation')
       ibl_plot_label2 = os.environ.get('IBL_PLOT_FCST_LABEL','Forecast')
       pb.plot_ibls(ibls_obs,lons_obs,ibl_plot_title,ibl_plot_outname,data2=ibls_fcst,
↪lon2=lons_fcst,
           label1=ibl_plot_label1,label2=ibl_plot_label2)


   # Run GIBL
   if ("GIBL" in steps_list_obs):
       if not ("IBL" in steps_list_obs):
           raise Exception('Must run observed IBLs before running GIBLs.')
       print('Computing Obs GIBLs')
```

(continues on next page)

(continued from previous page)

```
        gibls_obs = steps_obs.run_Calc_GIBL(ibls_obs,lons_obs)

    if ("GIBL" in steps_list_fcst):
        if not ("IBL" in steps_list_fcst):
            raise Exception('Must run Forecast IBLs before running GIBLs.')
        print('Computing Forecast GIBLs')
        gibls_fcst = steps_fcst.run_Calc_GIBL(ibls_fcst,lons_fcst)


    # Calc Blocks
    if ("CALCBLOCKS" in steps_list_obs):
        if not ("GIBL" in steps_list_obs):
            raise Exception('Must run observed GIBLs before calculating blocks.')
        print('Computing Obs Blocks')
        block_freq_obs = steps_obs.run_Calc_Blocks(ibls_obs,gibls_obs,lons_obs,daynum_obs)

    if ("CALCBLOCKS" in steps_list_fcst):
        if not ("GIBL" in steps_list_fcst):
            raise Exception('Must run Forecast GIBLs before calculating blocks.')
        print('Computing Forecast Blocks')
        block_freq_fcst = steps_fcst.run_Calc_Blocks(ibls_fcst,gibls_fcst,lons_fcst,daynum_
↪fcst)

    # Write out a Blocking MPR file if both obs and forecast blocking calculation performed
    if ("CALCBLOCKS" in steps_list_obs) and ("CALCBLOCKS" in steps_list_fcst):
        b_mpr_outdir = os.path.join(mpr_dir,'Blocks')
        if not os.path.exists(b_mpr_outdir):
            os.makedirs(b_mpr_outdir)
        # Print Blocks to output matched pair file
        modname = os.environ.get('MODEL_NAME','GFS')
        maskname = os.environ.get('MASK_NAME','FULL')
        blocks_outfile_prefix = os.path.join(b_mpr_outdir,'blocking_stat_'+modname)
        cbls_avg = np.nanmean(cbls_obs,axis=0)
        write_mpr_file(block_freq_obs,block_freq_fcst,cbls_avg,lons_obs,ibl_time_obs,ibl_
↪time_fcst,modname,
            'NA','Blocks','block','Z500','Blocks','block','Z500',maskname,'500',blocks_
↪outfile_prefix)


    # Plot Blocking Frequency
    if ("PLOTBLOCKS" in steps_list_obs):
        if not ("CALCBLOCKS" in steps_list_obs):
            raise Exception('Must compute observed blocks before plotting them.')
        print('Plotting Obs Blocks')
        blocking_plot_title = os.environ.get('OBS_BLOCKING_PLOT_TITLE','Obs Blocking␣
```

(continues on next page)

(continued from previous page)

```
↪Frequency')
        blocking_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_BLOCKING_PLOT_
↪OUTPUT_NAME','obs_Block_Freq'))
        pb.plot_blocks(block_freq_obs,gibls_obs,ibls_obs,lons_obs,blocking_plot_title,
↪blocking_plot_outname)
    if ("PLOTBLOCKS" in steps_list_fcst):
        if not ("CALCBLOCKS" in steps_list_fcst):
            raise Exception('Must compute forecast blocks before plotting them.')
        print('Plotting Forecast Blocks')
        blocking_plot_title = os.environ.get('FCST_BLOCKING_PLOT_TITLE','Forecast Blocking␣
↪Frequency')
        blocking_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_BLOCKING_PLOT_
↪OUTPUT_NAME','fcst_Block_Freq'))
        pb.plot_blocks(block_freq_fcst,gibls_fcst,ibls_fcst,lons_fcst,blocking_plot_title,
↪blocking_plot_outname)


if __name__ == "__main__":
    main()
```

Running METplus

This use case is run in the following ways:

1) Passing in UserScript_obsERA_obsOnly_Blocking.py then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_obsERA_obsOnly_Blocking.py -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_obsERA_obsOnly_Blocking.py:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_obsERA_obsOnly_Blocking.py
```

The following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/s2s_mid_lat/Blocking (relative to **OUTPUT_BASE**) and will contain output for the steps requested. This may include the regridded data, daily averaged files, running mean files, and anomaly files. In addition, output CBL, IBL, and Blocking frequency plots can be generated. The location of these output plots can be specified as BLOCKING_PLOT_OUTPUT_DIR. If it is not specified, plots will be sent to OUTPUT_BASE/plots. MET format matched pair output will also be generated for IBLs and blocks if a user runs these steps on both the model and observation data. The location the matched pair output can be specified as BLOCKING_MPR_OUTPUT_DIR. If it is not specified, plots will be sent to OUTPUT_BASE/mpr.

Keywords

Note:

- RegridDataPlaneToolUseCase
- PCPCombineToolUseCase
- S2SAppUseCase
- S2SMidLatAppUseCase
- NetCDFFileUseCase
- GRIB2FileUseCase
- METcalcpyUseCase
- METplotpyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s_mid_lat-UserScript_obsERA_obsOnly_Blocking.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.11.2 WeatherRegime Calculation: ERA RegridDataPlane, PcpCombine, and WeatherRegime python code

model_applications/ s2s_mid_lat/ UserScript_obsERA_obsOnly_WeatherRegime.py

Scientific Objective

To perform a weather regime analysis using 500 mb height data. There are 2 pre- processing steps, RegridDataPlane and PcpCombine, and 4 steps in the weather regime analysis, elbow, EOFs, K means, and the Time frequency. The elbow and K means steps begin with K means clustering. Elbow then computes the sum of squared distances for clusters 1 - 14 and draws a straight line from the sum of squared distance for the clusters. This helps determine the optimal cluster number by examining the largest difference between the curve and the straight line. The EOFs step is optional. It computes an empirical orthogonal function analysis. The K means step uses clustering to compute the frequency of occurrence and anomalies for each cluster to give the most common weather regimes. Then, the time frequency computes the frequency of each weather regime over a user specified time frame. Finally, stat_analysis can be run to compute an categorical analysis of the weather regime classification or an anomaly correlation of the time frequency data.

Datasets

- Forecast dataset: None.
- Observation dataset: ERA Reanlaysis 500 mb height.

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* numpy
* netCDF4
* datetime
* pylab
* scipy
* sklearn
* eofs
```

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case runs the weather regime driver script which runs the steps the user lists in STEPS_OBS. The possible steps are regridding, time averaging, creating a list of input files for the weather regime calculation, computing the elbow (ELBOW), plotting the elbow (PLOTELBOW), computing EOFs (EOF), plotting EOFs (PLOTEOF), computing K means (KMEANS), plotting the K means (PLOTKMEANS), computing a time frequency of weather regimes (TIMEFREQ) and plotting the time frequency (PLOTFREQ). All variables are set up in the UserScript .conf file. The pre- processing steps and stat_analysis are listed in the process list, and are formatted as follows:

PROCESS_LIST = RegridDataPlane(regrid_obs), PcpCombine(daily_mean_obs), UserScript(script_wr)

The other steps are listed in the [user_env_vars] section of the UserScript .conf file in the following format: OBS_STEPS = ELBOW+PLOTELBOW+EOF+PLOTEOF+KMEANS+PLOTKMEANS+TIMEFREQ+PLOTFREQ

METplus Workflow

The weather regime python code is run for each time for the forecast and observations data. This example loops by valid time. This version is set to only process the weather regime steps (ELBOW, PLOTELBOW, EOF, PLOTEOF, KMEANS, PLOTKMEANS, TIMEFREQ, PLOTFREQ) and stat_analysis, omitting the regridding, time averaging, and creating the file list pre-processing steps. However, the configurations for pre-processing are available for user reference.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/s2s_mid_lat/UserScript_obsERA_obsOnly_WeatherRegime.py. The file UserScript_obsERA_obsOnly_WeatherRegime.conf runs the python program and sets the variables for all steps of the Weather Regime use case including data paths.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s_mid_lat/
↪UserScript_obsERA_obsOnly_WeatherRegime.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###
```

(continues on next page)

(continued from previous page)

```
# All steps, including pre-processing:
# PROCESS_LIST = RegridDataPlane(regrid_obs), PcpCombine(daily_mean_obs), UserScript(script_
↪wr)
# Weather Regime Analysis only:
PROCESS_LIST = UserScript(script_wr)


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 1979120100
VALID_END = 2017022800
VALID_INCREMENT = 86400

LEAD_SEQ = 0

# Only Process DJF
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:0229"

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD


###
# RegridDataPlane(regrid_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Regridding Pre-Processing Step
[regrid_obs]

VALID_BEG = 1979120100
VALID_END = 2017022818
VALID_INCREMENT = 21600
```

(continues on next page)

(continued from previous page)

```
# REGRID_DATA_PLANE (Pre Processing Step 1), currently turned off

OBS_REGRID_DATA_PLANE_RUN = True

OBS_DATA_PLANE_ONCE_PER_FIELD = False

OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Z
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = P500
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = Z500

# Mask to use for regridding
# A 1 degree latitude/longitude grid running 24 to 54 degrees latitude
# and 230 to 300 degrees longitude
REGRID_DATA_PLANE_VERIF_GRID = latlon 71 31 54 230 -1.0 1.0

REGRID_DATA_PLANE_METHOD = BILIN

REGRID_DATA_PLANE_WIDTH = 2

OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mid_lat/UserScript_
↪fcstGFS_obsERA_WeatherRegime/ERA/OrigData
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪WeatherRegime/ERA/Regrid

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = {valid?fmt=%Y%m}/ei.oper.an.pl.regn128sc.{valid?fmt=%Y
↪%m%d%H}
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/Z500_6hourly_{init?fmt=%Y%m%d%H}_
↪NH.nc


###
# PCPCombine(daily_mean_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Daily Mean Pre-Processing Step
[daily_mean_obs]

VALID_BEG = 1979120118
VALID_END = 2017022818

OBS_PCP_COMBINE_RUN = True

OBS_PCP_COMBINE_METHOD = DERIVE
OBS_PCP_COMBINE_STAT_LIST = MEAN
```

(continues on next page)

(continued from previous page)

```
OBS_PCP_COMBINE_INPUT_ACCUMS = 6
OBS_PCP_COMBINE_INPUT_NAMES = Z500
OBS_PCP_COMBINE_INPUT_LEVELS = "(*,*)"
OBS_PCP_COMBINE_INPUT_OPTIONS = convert(x) = x / 9.81; set_attr_valid = "{valid?fmt=%Y%m%d_%H
↪%M%S?shift=-64800}";

# Convert height and derive mean over 24 hours
OBS_PCP_COMBINE_OUTPUT_ACCUM = 24
OBS_PCP_COMBINE_DERIVE_LOOKBACK = 24

OBS_PCP_COMBINE_OUTPUT_NAME = Z500

OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪WeatherRegime/ERA/Regrid
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪WeatherRegime/ERA/Daily

OBS_PCP_COMBINE_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/Z500_6hourly_{valid?fmt=%Y%m%d%H}_NH.nc
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = Z500_daily_{valid?fmt=%Y%m%d?shift=-64800}_NH.nc


# Variables for the Weather Regime code
[user_env_vars]
# Steps to Run
OBS_STEPS = ELBOW+PLOTELBOW+EOF+PLOTEOF+KMEANS+PLOTKMEANS+TIMEFREQ+PLOTFREQ

# Make OUTPUT_BASE Available to the script
SCRIPT_OUTPUT_BASE = {OUTPUT_BASE}

# Number of Seasons and Days per season that should be available
# The code will fill missing data, but requires the same number of days per
# season for each year.  You may need to omit leap days if February is part of
# the processing
NUM_SEASONS = 38
DAYS_PER_SEASON = 90

# Variable for the Z500 data
OBS_WR_VAR = Z500

# Weather Regime Number
OBS_WR_NUMBER = 6

# Number of clusters
OBS_NUM_CLUSTERS = 20
```

(continues on next page)

(continued from previous page)

```
# Number of principal components
OBS_NUM_PCS = 10

# Time (in timesteps) over which to compute weather regime frequencies
# i.e. if your data time step is days and you want to average over 7
# days, input 7
# Optional, only needed if you want to compute frequencies
OBS_WR_FREQ = 7

# Type, name and directory of Output File for weather regime classification
# Type options are text or netcdf
OBS_WR_OUTPUT_FILE_TYPE = text
OBS_WR_OUTPUT_FILE = obs_weather_regime_class
WR_OUTPUT_FILE_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_WeatherRegime

# Directory to send output plots
WR_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_WeatherRegime/plots/

# Elbow Plot Title and output file name
OBS_ELBOW_PLOT_TITLE = ERA Elbow Method For Optimal k
OBS_ELBOW_PLOT_OUTPUT_NAME = obs_elbow

# EOF plot output name and contour levels
OBS_EOF_PLOT_OUTPUT_NAME = obs_eof
EOF_PLOT_LEVELS = -50, -45, -40, -35, -30, -25, -20, -15, -10, -5,  0,  5, 10, 15, 20, 25,␣
↪30, 35, 40, 45, 50

# K means Plot Output Name and contour levels
OBS_KMEANS_PLOT_OUTPUT_NAME = obs_kmeans
KMEANS_PLOT_LEVELS = -80, -70, -60, -50, -40, -30, -20, -10, 0, 10, 20, 30, 40, 50, 60, 70,␣
↪80

# Frequency Plot title and output file name
OBS_FREQ_PLOT_TITLE = ERA Seasonal Cycle of WR Days/Week (1979-2017)
OBS_FREQ_PLOT_OUTPUT_NAME = obs_freq

# MPR file information
MASK_NAME = FULL
WR_MPR_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_WeatherRegime/mpr


###
# UserScript(script_wr) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
```

(continues on next page)

(continued from previous page)

```
###

# Run the Weather Regime Script
[script_wr]

# Template of filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mid_lat/UserScript_fcstGFS_
↪obsERA_WeatherRegime/ERA/Daily/Z500_daily_{valid?fmt=%Y%m%d}_NH.nc

# Name of the file containing the listing of input files
# The options are OBS_INPUT for observations or FCST_INPUT for forecast
# Or, set OBS_INPUT, FCST_INPUT if doing both and make sure the USER_SCRIPT_INPUT_TEMPLATE␣
↪is ordered:
# observation_template, forecast_template
USER_SCRIPT_INPUT_TEMPLATE_LABELS = OBS_INPUT

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_fcstGFS_obsERA_WeatherRegime/WeatherRegime_driver.py
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

See the following files for more information about the environment variables set in this configuration file.

parm/use_cases/met_tool_wrapper/RegridDataPlane/RegridDataPlane.py parm/use_cases/met_tool_wrapper/PCPCo
parm/use_cases/met_tool_wrapper/StatAnalysis/StatAnalysis.py

Python Scripts

This use case uses Python scripts to perform the blocking calculation

parm/use_cases/model_applications/s2s_mid_lat/UserScript_obsERA_obsOnly_WeatherRegime/WeatherRegime_drive
This script calls the requested steps in the blocking analysis for a forecast, observation, or both. The possible steps are computing the elbow, computing EOFs, and computing weather regimes using k means clustering.

metcalcpy/contributed/blocking_weather_regime/WeatherRegime.py: This script runs the requested steps, containing the code for computing the bend in the elbow, computing EOFs, and computing weather regimes using k means clustering. See the METcalcpy Weather Regime Calculation Script for more information.

metcalcpy/contributed/blocking_weather_regime//Blocking_WeatherRegime_util.py: This script contains functions used by both the blocking anwd weather regime analysis, including the code for determining which steps the user wants to run, and finding and reading the input files in the format from the output pre-processing steps. See the METcalcpy Utility script for more information.

```
#!/usr/bin/env python3
import sys
import os
import numpy as np
import netCDF4
import warnings

from metcalcpy.contributed.blocking_weather_regime.WeatherRegime import␣
↪WeatherRegimeCalculation
from metcalcpy.contributed.blocking_weather_regime.Blocking_WeatherRegime_util import parse_
↪steps, read_nc_met, write_mpr_file, reorder_fcst_regimes,reorder_fcst_regimes_correlate
from metplotpy.contributed.weather_regime import plot_weather_regime as pwr


def main():

    steps_list_fcst,steps_list_obs = parse_steps()

    if not steps_list_obs and not steps_list_fcst:
        warnings.warn('No processing steps requested for either the model or observations,')
        warnings.warn(' nothing will be run')
        warnings.warn('Set FCST_STEPS and/or OBS_STEPS in the [user_env_vars] section to␣
↪process data')


    ##########################################################################
    # Blocking Calculation and Plotting
    ##########################################################################
    # Set up the data
    steps_obs = WeatherRegimeCalculation('OBS')
    steps_fcst = WeatherRegimeCalculation('FCST')

    # Check to see if there is a plot directory
    oplot_dir = os.environ.get('WR_PLOT_OUTPUT_DIR','')
    obase = os.environ['SCRIPT_OUTPUT_BASE']
    if not oplot_dir:
        oplot_dir = os.path.join(obase,'plots')
    if not os.path.exists(oplot_dir):
        os.makedirs(oplot_dir)

     # Check to see if there is a mpr output directory
    mpr_outdir = os.environ.get('WR_MPR_OUTPUT_DIR','')
```

(continues on next page)

(continued from previous page)

```
    if not mpr_outdir:
        mpr_outdir = os.path.join(obase,'mpr')

     # Get number of seasons and days per season
    nseasons = int(os.environ['NUM_SEASONS'])
    dseasons = int(os.environ['DAYS_PER_SEASON'])

    # Grab the Daily text files
    obs_wr_filetxt = os.environ.get('METPLUS_FILELIST_OBS_INPUT','')
    fcst_wr_filetxt = os.environ.get('METPLUS_FILELIST_FCST_INPUT','')

    if ("ELBOW" in steps_list_obs) or ("EOF" in steps_list_obs) or ("KMEANS" in steps_list_
↪obs):
        with open(obs_wr_filetxt) as owl:
            obs_infiles = owl.read().splitlines()
        # Remove the first line if it's there
        if (obs_infiles[0] == 'file_list'):
            obs_infiles = obs_infiles[1:]
        if len(obs_infiles) != (nseasons*dseasons):
            raise Exception('Invalid Obs data; each year must contain the same date range to␣
↪calculate seasonal averages.')
        obs_invar = os.environ.get('OBS_WR_VAR','')
        z500_obs,lats_obs,lons_obs,timedict_obs = read_nc_met(obs_infiles,obs_invar,nseasons,
↪dseasons)
        z500_detrend_obs,z500_detrend_2d_obs = steps_obs.weights_detrend(lats_obs,lons_obs,
↪z500_obs)

    if ("ELBOW" in steps_list_fcst) or ("EOF" in steps_list_fcst) or("KMEANS" in steps_list_
↪fcst):
        with open(fcst_wr_filetxt) as fwl:
            fcst_infiles = fwl.read().splitlines()
        # Remove the first line if it's there
        if (fcst_infiles[0] == 'file_list'):
            fcst_infiles = fcst_infiles[1:]
        if len(fcst_infiles) != (nseasons*dseasons):
            raise Exception('Invalid Obs data; each year must contain the same date range to␣
↪calculate seasonal averages.')
        fcst_invar = os.environ.get('FCST_WR_VAR','')
        z500_fcst,lats_fcst,lons_fcst,timedict_fcst = read_nc_met(fcst_infiles,fcst_invar,
↪nseasons,dseasons)
        z500_detrend_fcst,z500_detrend_2d_fcst = steps_fcst.weights_detrend(lats_fcst,lons_
↪fcst,z500_fcst)


    if ("ELBOW" in steps_list_obs):
```

(continues on next page)

(continued from previous page)

```
        print('Running Obs Elbow')
        K_obs,d_obs,mi_obs,line_obs,curve_obs = steps_obs.run_elbow(z500_detrend_2d_obs)

    if ("ELBOW" in steps_list_fcst):
        print('Running Forecast Elbow')
        K_fcst,d_fcst,mi_fcst,line_fcst,curve_fcst = steps_fcst.run_elbow(z500_detrend_2d_
↪fcst)

    if ("PLOTELBOW" in steps_list_obs):
        if not ("ELBOW" in steps_list_obs):
            raise Exception('Must run observed Elbow before plotting observed elbow.')
        print('Creating Obs Elbow plot')
        elbow_plot_title = os.environ.get('OBS_ELBOW_PLOT_TITLE','Elbow Method For Optimal k
↪')
        elbow_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_ELBOW_PLOT_OUTPUT_
↪NAME','obs_elbow'))
        pwr.plot_elbow(K_obs,d_obs,mi_obs,line_obs,curve_obs,elbow_plot_title,elbow_plot_
↪outname)

    if ("PLOTELBOW" in steps_list_fcst):
        if not ("ELBOW" in steps_list_fcst):
            raise Exception('Must run forecast Elbow before plotting forecast elbow.')
        print('Creating Forecast Elbow plot')
        elbow_plot_title = os.environ.get('FCST_ELBOW_PLOT_TITLE','Elbow Method For Optimal k
↪')
        elbow_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_ELBOW_PLOT_OUTPUT_
↪NAME','fcst_elbow'))
        pwr.plot_elbow(K_fcst,d_fcst,mi_fcst,line_fcst,curve_fcst,elbow_plot_title,elbow_
↪plot_outname)


    if ("EOF" in steps_list_obs):
        print('Running Obs EOF')
        eof_obs,pc_obs,wrnum_obs,variance_fractions_obs = steps_obs.Calc_EOF(z500_obs)
        z500_detrend_2d_obs = steps_obs.reconstruct_heights(eof_obs,pc_obs,z500_detrend_2d_
↪obs.shape)

    if ("EOF" in steps_list_fcst):
        print('Running Forecast EOF')
        eof_fcst,pc_fcst,wrnum_fcst,variance_fractions_fcst = steps_fcst.Calc_EOF(z500_fcst)
        z500_detrend_2d_fcst = steps_fcst.reconstruct_heights(eof_fcst,pc_fcst,z500_detrend_
↪2d_fcst.shape)

    if ("PLOTEOF" in steps_list_obs):
        if not ("EOF" in steps_list_obs):
```

(continues on next page)

(continued from previous page)

```
            raise Exception('Must run observed EOFs before plotting observed EOFs.')
        print('Plotting Obs EOFs')
        pltlvls_str = os.environ['EOF_PLOT_LEVELS'].split(',')
        pltlvls = [float(pp) for pp in pltlvls_str]
        eof_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_EOF_PLOT_OUTPUT_NAME',
↪'obs_eof'))
        pwr.plot_eof(eof_obs,wrnum_obs,variance_fractions_obs,lons_obs,lats_obs,eof_plot_
↪outname,pltlvls)

    if ("PLOTEOF" in steps_list_fcst):
        if not ("EOF" in steps_list_fcst):
            raise Exception('Must run forecast EOFs before plotting forecast EOFs.')
        print('Plotting Forecast EOFs')
        pltlvls_str = os.environ['EOF_PLOT_LEVELS'].split(',')
        pltlvls = [float(pp) for pp in pltlvls_str]
        eof_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_EOF_PLOT_OUTPUT_NAME',
↪'fcst_eof'))
        pwr.plot_eof(eof_fcst,wrnum_fcst,variance_fractions_fcst,lons_fcst,lats_fcst,eof_
↪plot_outname,pltlvls)


    if ("KMEANS" in steps_list_obs):
        print('Running Obs K Means')
        kmeans_obs,wrnum_obs,perc_obs,wrc_obs= steps_obs.run_K_means(z500_detrend_2d_obs,
↪timedict_obs,z500_obs.shape)
        steps_obs.write_K_means_file(timedict_obs,wrc_obs)

    if ("KMEANS" in steps_list_fcst):
        print('Running Forecast K Means')
        kmeans_fcst,wrnum_fcst,perc_fcst,wrc_fcst = steps_fcst.run_K_means(z500_detrend_2d_
↪fcst,timedict_fcst,
            z500_fcst.shape)
        reorder_fcst = os.environ.get('REORDER_FCST','False').lower()
        reorder_fcst_manual = os.environ.get('REORDER_FCST_MANUAL','False').lower()
        if (reorder_fcst == 'true') and ("KMEANS" in steps_list_obs):
            kmeans_fcst,perc_fcst,wrc_fcst = reorder_fcst_regimes_correlate(kmeans_obs,
↪kmeans_fcst,perc_fcst,wrc_fcst,wrnum_fcst)
        if reorder_fcst_manual == 'true':
            fcst_order_str = os.environ['FCST_ORDER'].split(',')
            fcst_order = [int(fo) for fo in fcst_order_str]
            kmeans_fcst,perc_fcst,wrc_fcst = reorder_fcst_regimes(kmeans_fcst,perc_fcst,wrc_
↪fcst,wrnum_fcst,fcst_order)
        steps_fcst.write_K_means_file(timedict_fcst,wrc_fcst)
```

(continues on next page)

(continued from previous page)

```
        # Write matched pair output for weather regime classification
        modname = os.environ.get('MODEL_NAME','GFS')
        maskname = os.environ.get('MASK_NAME','FULL')
        mpr_full_outdir = os.path.join(mpr_outdir,'WeatherRegime')
        wr_outfile_prefix = os.path.join(mpr_full_outdir,'weather_regime_stat_'+modname)
        wrc_obs_mpr = wrc_obs[:,:,np.newaxis]
        wrc_fcst_mpr = wrc_fcst[:,:,np.newaxis]
        if not os.path.exists(mpr_full_outdir):
            os.makedirs(mpr_full_outdir)
        write_mpr_file(wrc_obs_mpr,wrc_fcst_mpr,[0.0],[0.0],timedict_obs,timedict_fcst,
↪modname,'NA',
            'WeatherRegimeClass','class','Z500','WeatherRegimeClass','class','Z500',maskname,
↪'500',wr_outfile_prefix)

    if ("PLOTKMEANS" in steps_list_obs):
        if not ("KMEANS" in steps_list_obs):
            raise Exception('Must run observed Kmeans before plotting observed Kmeans.')
        print('Plotting Obs K Means')
        pltlvls_str = os.environ['KMEANS_PLOT_LEVELS'].split(',')
        pltlvls = [float(pp) for pp in pltlvls_str]
        kmeans_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_KMEANS_PLOT_OUTPUT_
↪NAME','obs_kmeans'))
        pwr.plot_K_means(kmeans_obs,wrnum_obs,lons_obs,lats_obs,perc_obs,kmeans_plot_outname,
↪pltlvls)

    if ("PLOTKMEANS" in steps_list_fcst):
        if not ("KMEANS" in steps_list_fcst):
            raise Exception('Must run forecast Kmeans before plotting forecast Kmeans.')
        print('Plotting Forecast K Means')
        pltlvls_str = os.environ['KMEANS_PLOT_LEVELS'].split(',')
        pltlvls = [float(pp) for pp in pltlvls_str]
        kmeans_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_KMEANS_PLOT_OUTPUT_
↪NAME','fcst_kmeans'))
        pwr.plot_K_means(kmeans_fcst,wrnum_fcst,lons_fcst,lats_fcst,perc_fcst,kmeans_plot_
↪outname,pltlvls)


    if ("TIMEFREQ" in steps_list_obs):
        if not ("KMEANS" in steps_list_obs):
            raise Exception('Must run observed Kmeans before running frequencies.')
        wrfreq_obs,dlen_obs,ts_diff_obs = steps_obs.compute_wr_freq(wrc_obs)

    if ("TIMEFREQ" in steps_list_fcst):
        if not ("KMEANS" in steps_list_fcst):
            raise Exception('Must run forecast Kmeans before running frequencies.')
```

(continues on next page)

(continued from previous page)

```
        wrfreq_fcst,dlen_fcst,ts_diff_fcst = steps_fcst.compute_wr_freq(wrc_fcst)

    if ("TIMEFREQ" in steps_list_obs) and ("TIMEFREQ" in steps_list_fcst):
        # Write matched pair output for frequency of each weather regime
        modname = os.environ.get('MODEL_NAME','GFS')
        maskname = os.environ.get('MASK_NAME','FULL')
        mpr_full_outdir = os.path.join(mpr_outdir,'freq')
        timedict_obs_mpr = {'init':timedict_obs['init'][:,ts_diff_obs-1:],
            'valid':timedict_obs['valid'][:,ts_diff_obs-1:],'lead':timedict_obs['lead'][:,ts_
↪diff_obs-1:]}
        timedict_fcst_mpr = {'init':timedict_fcst['init'][:,ts_diff_fcst-1:],
            'valid':timedict_fcst['valid'][:,ts_diff_fcst-1:],'lead':timedict_fcst['lead'][:,
↪ts_diff_fcst-1:]}
        wrfreq_obs_mpr = wrfreq_obs[:,:,:,np.newaxis]
        wrfreq_fcst_mpr = wrfreq_fcst[:,:,:,np.newaxis]
        if not os.path.exists(mpr_full_outdir):
            os.makedirs(mpr_full_outdir)
        for wrn in np.arange(wrnum_obs):
            wr_outfile_prefix = os.path.join(mpr_full_outdir,'weather_regime'+str(wrn+1).
↪zfill(2)+'_freq_stat_'+modname)
            write_mpr_file(wrfreq_obs_mpr[wrn,:,:,:],wrfreq_fcst_mpr[wrn,:,:,:],[0.0],[0.0],
↪timedict_obs,
                timedict_fcst,modname,str(wrn+1).zfill(2),'WeatherRegimeFreq','percent','Z500
↪','WeatherRegimeFreq',
                'percent','Z500',maskname,'500',wr_outfile_prefix)

    if ("PLOTFREQ" in steps_list_obs):
        if not ("TIMEFREQ" in steps_list_obs):
            raise Exception('Must run observed Frequency calculation before plotting the␣
↪frequencies.')
        freq_plot_title = os.environ.get('OBS_FREQ_PLOT_TITLE','Seasonal Cycle of WR Days/
↪Week')
        freq_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_FREQ_PLOT_OUTPUT_NAME
↪','obs_freq'))
        # Compute mean
        wrmean_obs = np.nanmean(wrfreq_obs,axis=1)
        pwr.plot_wr_frequency(wrmean_obs,wrnum_obs,dlen_obs,freq_plot_title,freq_plot_
↪outname)

    if ("PLOTFREQ" in steps_list_fcst):
        if not ("TIMEFREQ" in steps_list_fcst):
            raise Exception('Must run forecast Frequency calculation before plotting the␣
↪frequencies.')
        freq_plot_title = os.environ.get('FCST_FREQ_PLOT_TITLE','Seasonal Cycle of WR Days/
↪Week')
```

(continues on next page)

(continued from previous page)

```
        freq_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_FREQ_PLOT_OUTPUT_NAME
↪','fcst_freq'))
        # Compute mean
        wrmean_fcst = np.nanmean(wrfreq_fcst,axis=1)
        pwr.plot_wr_frequency(wrmean_fcst,wrnum_fcst,dlen_fcst,freq_plot_title,freq_plot_
↪outname)


if __name__ == "__main__":
    main()
```

Running METplus

This use case is run in the following ways:

1) Passing in UserScript_obsERA_obsOnly_WeatherRegime.py then a user-specific system configuration file:

```
master_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_obsERA_obsOnly_WeatherRegime.py -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_obsERA_obsOnly_WeatherRegime.py:

```
master_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_obsERA_obsOnly_WeatherRegime.py
```

The following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/s2s_mid_lat/WeatherRegime (relative to **OUTPUT_BASE**) and will contain output for the steps requested. This may include the regridded data, daily averaged files, a text file containing the list of input files, and text files for the weather regime classification and time frequency (if KMEANS and TIMEFREQ are run for both the forecast and observation data). In addition, output elbow, EOF, and Kmeans weather regime plots can be generated. The location of these output plots can be specified as WR_OUTPUT_DIR. If it is not specified, plots will be sent to {OUTPUT_BASE}/plots. The output location for the matched pair files can be specified as WR_MPR_OUTPUT_DIR. If it is not specified, it will be sent to {OUTPUT_BASE}/mpr. The output weather regime text or netCDF file location is set in WR_OUTPUT_FILE_DIR. If this is not specified, the output text/netCDF file will be sent to {OUTPUT_BASE}. The stat_analysis contingency table statistics and anomaly correlation files will be sent to the locations given in STAT_ANALYSIS_OUTPUT_DIR for their respective configuration sections.

Keywords

Note:

- RegridDataPlaneToolUseCase
- PCPCombineToolUseCase
- S2SAppUseCase
- S2SMidLatAppUseCase
- NetCDFFileUseCase
- GRIB2FileUseCase
- METcalcpyUseCase
- METplotpyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s_mid_lat-UserScript_obsERA_obsOnly_WeatherRegime.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.11.3 Blocking Calculation: GFS and ERA RegridDataPlane, PcpCombine, and Blocking python code

model_applications/ s2s_mid_lat/ UserScript_fcstGFS_obsERA_Blocking.py

Scientific Objective

To compute the Central Blocking Latitude, Instantaneousy blocked latitudes, Group Instantaneousy blocked latitudes, and the frequency of atmospheric blocking using the Pelly-Hoskins Method. After these are computed, contingency table statistics are computed on the Instantaneous blocked latitudes and blocks using stat_analysis.

Datasets

- Forecast dataset: GFS Forecast 500 mb height.
- Observation dataset: ERA Reanlaysis 500 mb height.

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* numpy
* netCDF4
* datetime
* bisect
* scipy
```

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case runs the blocking driver script which runs the steps the user lists in STEPS_OBS. The possible steps are regridding, time averaging, computing a running mean, computing anomalies, computing CBLs (CBL), plotting CBLs (PLOTCBL), computing IBLs (IBL), plotting IBL frequency (PLOTIBL), computing GIBLs (GIBL), computing blocks (CALCBLOCKS), plotting the blocking frequency (PLOTBLOCKS) and using stat_analysis to compute statistics on the IBL or blocking results. Regridding, time averaging, running means, anomaloies, and stat_analysis are set up in the UserScript .conf file and

are formatted as follows: PROCESS_LIST = RegridDataPlane(regrid_fcst), RegridDataPlane(regrid_obs), PcpCombine(daily_mean_fcst), PcpCombine(daily_mean_obs), PcpCombine(running_mean_obs), PcpCombine(anomaly_obs), UserScript(create_cbl_filelist), UserScript(script_blocking), StatAnalysis(sanal_ibls), StatAnalysis(sanal_blocks)

The other steps are listed in the Blocking .conf file and are formatted as follows: FCST_STEPS = CBL+IBL+PLOTIBL+GILB+CALCBLOCKS+PLOTBLOCKS OBS_STEPS = CBL+PLOTCBL+IBL+PLOTIBL+GILB+CALCBLOCKS+PLOTBLOCKS

METplus Workflow

The blocking python code is run for each time for the forecast and observations data. This example loops by init time for the model pre-processing, and valid time for the other steps. This version is set to only process the blocking steps (CBL, PLOTCBL, IBL, PLOTIBL) and stat_analysis, omitting the regridding, time averaging, running mean, and anomaly pre-processing steps. However, the configurations for pre-processing are available for user reference.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking.py. The file UserScript_fcstGFS_obsERA_Blocking.conf runs the python program, and the variables for all steps of the Blocking calculation are given in the [user_env_vars] section of the .conf file.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s_mid_lat/
↪UserScript_fcstGFS_obsERA_Blocking.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript(create_cbl_filelist), UserScript(script_blocking),␣
↪StatAnalysis(sanal_ibls), StatAnalysis(sanal_blocks)

# use this process list if pre-processing steps are needed
# PROCESS_LIST = RegridDataPlane(regrid_fcst), RegridDataPlane(regrid_obs), PcpCombine(daily_
↪mean_fcst), PcpCombine(daily_mean_obs), PcpCombine(running_mean_obs), PcpCombine(anomaly_
↪obs), UserScript(create_cbl_filelist), UserScript(script_blocking)
```

(continues on next page)

(continued from previous page)

```
###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2000120100
VALID_END = 2017022800
VALID_INCREMENT = 86400

LEAD_SEQ = 0

# Only Process DJF
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:0229"


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

OBS_ANOM_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪Blocking/ERA/Anomaly
OBS_ANOM_INPUT_TEMPLATE = Z500_anomaly_{valid?fmt=%Y%m%d}_NH.nc

OBS_ANOM_OUTPUT_DIR = {OBS_ANOM_INPUT_DIR}
OBS_ANOM_OUTPUT_TEMPLATE = ERA_anom_files_lead{lead?fmt=%HHH}.txt

OBS_AVE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪Blocking/ERA/Daily
OBS_AVE_INPUT_TEMPLATE = Z500_daily_{valid?fmt=%Y%m%d}_NH.nc

OBS_AVE_OUTPUT_DIR = {OBS_AVE_INPUT_DIR}
OBS_AVE_OUTPUT_TEMPLATE = ERA_daily_files_lead{lead?fmt=%HHH}.txt
```

(continues on next page)

(continued from previous page)

```
FCST_AVE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪Blocking/GFS/Daily
FCST_AVE_INPUT_TEMPLATE = Z500_daily_{init?fmt=%Y%m%d}_{lead?fmt=%HHH}_NH.nc

FCST_AVE_OUTPUT_DIR = {FCST_AVE_INPUT_DIR}
FCST_AVE_OUTPUT_TEMPLATE = GFS_daily_files_lead{lead?fmt=%HHH}.txt


###
# RegridDataPlane(regrid_fcst) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Forecast Regridding to 1 degree using regrid_data_plane
[regrid_fcst]

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2000120100
INIT_END = 2017022800
INIT_INCREMENT = 86400

LEAD_SEQ = 24

# REGRID_DATA_PLANE (Step 1)

FCST_REGRID_DATA_PLANE_RUN = True

FCST_DATA_PLANE_ONCE_PER_FIELD = False

FCST_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Z500
FCST_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = P500
FCST_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = Z500

REGRID_DATA_PLANE_VERIF_GRID = latlon 360 90 89 0 -1.0 1.0

REGRID_DATA_PLANE_METHOD = BILIN

REGRID_DATA_PLANE_WIDTH = 2

FCST_REGRID_DATA_PLANE_INPUT_DIR = /gpfs/fs1/p/ral/jntp/GMTB/Phys_Test_FV3GFSv2/POST/suite1/
FCST_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪Blocking/FV3GFS/Regrid
```

(continues on next page)

(continued from previous page)

```
FCST_REGRID_DATA_PLANE_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/gfs.t00z.pgrb2.0p25.f{lead?fmt=
↪%HHH}
FCST_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/Z500_3hourly_{init?fmt=%Y%m%d%H}
↪_{lead?fmt=%HHH}_NH.nc


###
# RegridDataPlane(regrid_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Observation Regridding to 1 degree using regrid_data_plane
[regrid_obs]

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 1979120100
VALID_END = 2017022818
VALID_INCREMENT = 21600

LEAD_SEQ = 0

OBS_REGRID_DATA_PLANE_RUN = True

OBS_DATA_PLANE_ONCE_PER_FIELD = False

OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Z
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = P500
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = Z500

REGRID_DATA_PLANE_VERIF_GRID = latlon 360 90 89 0 -1.0 1.0

REGRID_DATA_PLANE_METHOD = BILIN

REGRID_DATA_PLANE_WIDTH = 2

OBS_REGRID_DATA_PLANE_INPUT_DIR = /gpfs/fs1/collections/rda/data/ds627.0/ei.oper.an.pl
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪Blocking/ERA/Regrid

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = {valid?fmt=%Y%m}/ei.oper.an.pl.regn128sc.{valid?fmt=%Y
↪%m%d%H}
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/Z500_6hourly_{init?fmt=%Y%m%d%H}_
↪NH.nc
```

(continues on next page)

(continued from previous page)

```
###
# PCPCombine(daily_mean_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Perform a sum over the 4 daily times that have been regridded using pcp_combine
# 00, 06, 12, 18 UTC
[daily_mean_obs]

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 1979120118
VALID_END = 2017022818
VALID_INCREMENT = 86400

OBS_PCP_COMBINE_RUN = True

OBS_PCP_COMBINE_METHOD = DERIVE
OBS_PCP_COMBINE_STAT_LIST = MEAN

OBS_PCP_COMBINE_INPUT_ACCUMS = 6
OBS_PCP_COMBINE_INPUT_NAMES = Z500
OBS_PCP_COMBINE_INPUT_LEVELS = "(*,*)"
OBS_PCP_COMBINE_INPUT_OPTIONS = convert(x) = x / 9.81; set_attr_valid = "{valid?fmt=%Y%m%d_%H
↪%M%S?shift=-64800}";

OBS_PCP_COMBINE_OUTPUT_NAME = Z500
OBS_PCP_COMBINE_OUTPUT_ACCUM = 24
OBS_PCP_COMBINE_DERIVE_LOOKBACK = 24

OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/ERA/
↪Regrid
OBS_PCP_COMBINE_OUTPUT_DIR = {OBS_AVE_INPUT_DIR}

# Input ERA Interim
OBS_PCP_COMBINE_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/Z500_6hourly_{valid?fmt=%Y%m%d%H}_NH.nc
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = {OBS_AVE_INPUT_TEMPLATE}


###
# PCPCombine(running_mean_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###
```

(continues on next page)

(continued from previous page)

```
# Perform a 5 day running mean on the data using pcp_combine
[running_mean_obs]

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 1979120100
VALID_END = 2017022800
VALID_INCREMENT = 86400

# Add the first/last 2 days to the skip times to compute the running mean
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:1201,1202,0227,0228,0229"

OBS_PCP_COMBINE_RUN = TRUE
OBS_PCP_COMBINE_METHOD = DERIVE

OBS_PCP_COMBINE_STAT_LIST = MEAN

OBS_PCP_COMBINE_INPUT_ACCUMS = 24
OBS_PCP_COMBINE_INPUT_NAMES = Z500
OBS_PCP_COMBINE_INPUT_LEVELS = "(*,*)"
OBS_PCP_COMBINE_INPUT_OPTIONS = set_attr_valid = "{valid?fmt=%Y%m%d_%H%M%S?shift=-172800}";

OBS_PCP_COMBINE_OUTPUT_NAME = Z500

#  Running mean is 5 days
OBS_PCP_COMBINE_OUTPUT_ACCUM = 120
OBS_PCP_COMBINE_DERIVE_LOOKBACK = 120


OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/ERA/
↪Daily
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/
↪ERA/Rmean5d

OBS_PCP_COMBINE_INPUT_TEMPLATE = Z500_daily_{valid?fmt=%Y%m%d}_NH.nc
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = Z500_5daymean_{valid?fmt=%Y%m%d?shift=-172800}_NH.nc


###
# PCPCombine(anomaly_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Compute anomalies using the daily means and 5 day running mean using pcp_combine
[anomaly_obs]
```

(continues on next page)

(continued from previous page)

```
LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 1979120100
VALID_END = 2017022800
VALID_INCREMENT = 86400

LEAD_SEQ = 0

# Add the first/last 2 days to the skip times to compute the running mean
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:1201,1202,0227,0228,0229"


OBS_PCP_COMBINE_RUN = True
OBS_PCP_COMBINE_METHOD = USER_DEFINED

OBS_PCP_COMBINE_COMMAND = -subtract {OBS_PCP_COMBINE_INPUT_DIR}/Daily/Z500_daily_{valid?fmt=
↪%Y%m%d}_NH.nc {OBS_PCP_COMBINE_INPUT_DIR}/Rmean5d/Z500_5daymean_{valid?fmt=%Y%m%d}_NH.nc -
↪field 'name="Z500"; level="(*,*)";'

OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/ERA
OBS_PCP_COMBINE_OUTPUT_DIR = {OBS_ANOM_INPUT_DIR}

OBS_PCP_COMBINE_INPUT_TEMPLATE = Z500_daily_{valid?fmt=%Y%m%d}_NH.nc
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = {OBS_ANOM_INPUT_TEMPLATE}


###
# UserScript(create_cbl_filelist) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# This is run separately since it has different start/end times
[create_cbl_filelist]

# Skip the days on the edges that are not available due to the running mean
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:1201,0229"

# Find the files for each lead time
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

# Valid Begin and End Times for the CBL File Climatology
VALID_BEG = 1979120100
VALID_END = 2017022800
VALID_INCREMENT = 86400
```

(continues on next page)

(continued from previous page)

```
LEAD_SEQ = 0

USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mid_lat/UserScript_fcstGFS_
↪obsERA_Blocking/ERA/Anomaly/Z500_anomaly_{valid?fmt=%Y%m%d}_NH.nc

# Name of the file containing the listing of input files
USER_SCRIPT_INPUT_TEMPLATE_LABELS = OBS_CBL_INPUT

# Placeholder command just to build the file list
# This just states that it's building the file list
USER_SCRIPT_COMMAND = echo Populated file list for CBL Input


[user_env_vars]
# Obs and/or Forecast
FCST_STEPS = CBL+IBL+PLOTIBL+GIBL+CALCBLOCKS+PLOTBLOCKS
OBS_STEPS = CBL+PLOTCBL+IBL+PLOTIBL+GIBL+CALCBLOCKS+PLOTBLOCKS

# Number of Seasons and Days per season that should be available
# The code will fill missing data, but requires the same number of days per
# season for each year.  You may need to omit leap days if February is part of
# the processing
CBL_NUM_SEASONS = 38
IBL_NUM_SEASONS = 17
DAYS_PER_SEASON = 89

# Use the obs climatology for the calculation of CBL data because the forecast
# does not have a long enough data history.  Set to False if not wanting to
# use the obs
USE_CBL_OBS = True

# Variable Name for the Z500 anomaly data to read in to the blocking python code
OBS_BLOCKING_ANOMALY_VAR = Z500_ANA

# Variable for the Z500 data
FCST_BLOCKING_VAR = Z500_P500
OBS_BLOCKING_VAR = Z500

# Number of model grid points used for a moving average
# Must be odd
FCST_SMOOTHING_PTS = 9
OBS_SMOOTHING_PTS = {FCST_SMOOTHING_PTS}

# Lat Delta, to allow for offset from the Central Blocking Latitude
FCST_LAT_DELTA = -5,0,5
```

(continues on next page)

(continued from previous page)

```
OBS_LAT_DELTA = {FCST_LAT_DELTA}

# Meridional Extent of blocks (NORTH_SOUTH_LIMITS/2)
FCST_NORTH_SOUTH_LIMITS = 30
OBS_NORTH_SOUTH_LIMITS = {FCST_NORTH_SOUTH_LIMITS}

# Maximum number of grid points between IBLs for everything in between to be included as an␣
↪IBL
FCST_IBL_DIST = 7
OBS_IBL_DIST = {FCST_IBL_DIST}

# Number of grid points in and IBL to make a GIBL
FCST_IBL_IN_GIBL = 15
OBS_IBL_IN_GIBL = {FCST_IBL_IN_GIBL}

# Number of grid points that must overlap across days for a GIBL
FCST_GIBL_OVERLAP = 10
OBS_GIBL_OVERLAP = {FCST_GIBL_OVERLAP}

# Time duration in days needed for a block
FCST_BLOCK_TIME = 5
OBS_BLOCK_TIME = {FCST_BLOCK_TIME}

# Number of grid points a block must travel to terminate
FCST_BLOCK_TRAVEL = 45
OBS_BLOCK_TRAVEL = {FCST_BLOCK_TRAVEL}

# Method to compute blocking.  Currently, the only option is 'PH' for the
# Pelly-Hoskins Method
FCST_BLOCK_METHOD = PH
OBS_BLOCK_METHOD = {FCST_BLOCK_METHOD}

# Location of output MPR files
BLOCKING_MPR_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/mpr

# Plots Output Dir
BLOCKING_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/plots

#CBL plot title and output namename
OBS_CBL_PLOT_MTHSTR = DJF
OBS_CBL_PLOT_OUTPUT_NAME = ERA_CBL_avg

# IBL plot title and output name
IBL_PLOT_TITLE = DJF Instantaneous Blocked Longitude
IBL_PLOT_OUTPUT_NAME = FV3_ERA_IBL_Freq_DJF
```

(continues on next page)

(continued from previous page)

```
# IBL plot legend for forecast and obs
IBL_PLOT_OBS_LABEL = ERA Reanalysis
IBL_PLOT_FCST_LABEL = GEFS


###
# UserScript(script_blocking) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Run the Blocking Analysis Script
[script_blocking]
# Timing Information
LEAD_SEQ = 24

# Skip the days on the edges that are not available due to the running mean
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:1201,0229"

# Run the user script once for each lead
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

# Template of filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mid_lat/UserScript_fcstGFS_
↪obsERA_Blocking/ERA/Daily/Z500_daily_{valid?fmt=%Y%m%d}_NH.nc,{INPUT_BASE}/model_
↪applications/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/GFS/Daily/Z500_{init?fmt=%Y%m
↪%d}_{lead?fmt=%HHH}_NH.nc

# Name of the file containing the listing of input files
# The options are OBS_CBL_INPUT, FCST_CBL_INPUT, OBS_IBL_INPUT, and FCST_IBL_INPUT
# *** Make sure the order is the same as the order of templates listed in USER_SCRIPT_INPUT_
↪TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS = OBS_IBL_INPUT, FCST_IBL_INPUT

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_fcstGFS_obsERA_Blocking/Blocking_driver.py


###
# StatAnalysis(sanal_ibls) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#statanalysis
###

# Stat Analysis for the IBLs
```

(continues on next page)

(continued from previous page)

```
[sanal_ibls]

VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20001201
VALID_END = 20170228

STAT_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/STATAnalysisConfig_wrapped

MODEL1 = GFS
MODEL1_OBTYPE = ADPUPA

# stat_analysis job info
STAT_ANALYSIS_JOB_NAME = aggregate_stat
# if using -dump_row, put in JOBS_ARGS "-dump_row [dump_row_file]"
# if using -out_stat, put in JOBS_ARGS "-out_stat [out_stat_file]"
# METplus will fill in filename
STAT_ANALYSIS_JOB_ARGS = -out_line_type CTS -out_thresh ==1 -out_stat [out_stat_file]

MODEL_LIST = {MODEL1}
FCST_LEAD_LIST = 24
LINE_TYPE_LIST = MPR

GROUP_LIST_ITEMS = MODEL_LIST
LOOP_LIST_ITEMS = FCST_LEAD_LIST

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪Blocking/mpr/IBL

STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking

MODEL1_STAT_ANALYSIS_OUT_STAT_TEMPLATE = {model?fmt=%s}_ERA_IBLS_{lead?fmt=%H%M%S}L_CTS_CNT.
↪stat


###
# StatAnalysis(sanal_blocks) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#statanalysis
###

# Stat Analysis for the Blocks
[sanal_blocks]

VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20001201
VALID_END = 20170228
```

(continues on next page)

(continued from previous page)

```
STAT_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/STATAnalysisConfig_wrapped

MODEL1 = GFS
MODEL1_OBTYPE = ADPUPA

STAT_ANALYSIS_JOB_NAME = aggregate_stat
STAT_ANALYSIS_JOB_ARGS = -out_line_type CTS -out_thresh ==1 -out_stat [out_stat_file]

MODEL_LIST = {MODEL1}
FCST_LEAD_LIST = 24
LINE_TYPE_LIST = MPR

GROUP_LIST_ITEMS = MODEL_LIST
LOOP_LIST_ITEMS = FCST_LEAD_LIST

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪Blocking/mpr/Blocks

STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking

MODEL1_STAT_ANALYSIS_OUT_STAT_TEMPLATE = {model?fmt=%s}_ERA_Blocks_{lead?fmt=%H%M%S}L_CTS.
↪stat
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

See the following files for more information about the environment variables set in this configuration file.

parm/use_cases/met_tool_wrapper/RegridDataPlane/RegridDataPlane.py parm/use_cases/met_tool_wrapper/PCPCo
parm/use_cases/met_tool_wrapper/PCPCombine/PCPCOmbine_subtract.py parm/use_cases/met_tool_wrapper/StatA

Python Scripts

This use case uses Python scripts to perform the blocking calculation

parm/use_cases/model_applications/s2s_mid_lat/UserScript_fcstGFS_obsERA_Blocking/Blocking_driver.py: This script calls the requested steps in the blocking analysis for a forecast, observation, or both. The possible steps are computing CBLs, plotting CBLs, computing IBLs, plotting IBLs, computing GIBLs, computing blocks, and plotting blocks.

metcalcpy/contributed/blocking_weather_regime/Blocking.py: This script runs the requested steps, containing the code for computing CBLs, computing IBLs, computing GIBLs, and computing blocks. See the METcalcpy Blocking Calculation Script for more information.

metcalcpy/contributed/blocking_weather_regime/Blocking_WeatherRegime_util.py: This script contains functions used by both the blocking anwd weather regime analysis, including the code for determining which steps the user wants to run, and finding and reading the input files in the format from the output pre-processing steps. See the METcalcpy Utility script for more information.

```
#!/usr/bin/env python3
import sys
import os
import numpy as np
import datetime
import netCDF4
import warnings

from metcalcpy.contributed.blocking_weather_regime.Blocking import BlockingCalculation
from metcalcpy.contributed.blocking_weather_regime.Blocking_WeatherRegime_util import parse_
↪steps, write_mpr_file
from metplotpy.contributed.blocking_s2s import plot_blocking as pb
from metplotpy.contributed.blocking_s2s.CBL_plot import create_cbl_plot


def main():

    steps_list_fcst,steps_list_obs = parse_steps()

    if not steps_list_obs and not steps_list_fcst:
        warnings.warn('No processing steps requested for either the model or observations,')
        warnings.warn(' nothing will be run')
        warnings.warn('Set FCST_STEPS and/or OBS_STEPS in the [user_env_vars] section to␣
↪process data')


    ######################################################################
    # Blocking Calculation and Plotting
    ######################################################################
    # Set up the data
```

(continues on next page)

(continued from previous page)

```
    steps_fcst = BlockingCalculation('FCST')
    steps_obs = BlockingCalculation('OBS')

    # Check to see if there is a plot directory
    oplot_dir = os.environ.get('BLOCKING_PLOT_OUTPUT_DIR','')
    if not oplot_dir:
        obase = os.environ['SCRIPT_OUTPUT_BASE']
        oplot_dir = os.path.join(obase,'plots')
    if not os.path.exists(oplot_dir):
        os.makedirs(oplot_dir)

    # Check to see if there is a mpr output directory
    mpr_dir = os.environ.get('BLOCKING_MPR_OUTPUT_DIR','')
    if not mpr_dir:
        obase = os.environ['SCRIPT_OUTPUT_BASE']
        mpr_dir = os.path.join(obase,'mpr')

    # Check to see if CBL's are used from an obs climatology
    use_cbl_obs = os.environ.get('USE_CBL_OBS','False').lower()

    # Get the days per season
    dseasons = int(os.environ['DAYS_PER_SEASON'])

    # Grab the Anomaly (CBL) text files
    obs_cbl_filetxt = os.environ.get('METPLUS_FILELIST_OBS_CBL_INPUT','')
    fcst_cbl_filetxt = os.environ.get('METPLUS_FILELIST_FCST_CBL_INPUT','')

    # Grab the Daily (IBL) text files
    obs_ibl_filetxt = os.environ.get('METPLUS_FILELIST_OBS_IBL_INPUT','')
    fcst_ibl_filetxt = os.environ.get('METPLUS_FILELIST_FCST_IBL_INPUT','')


    # Calculate Central Blocking Latitude
    if ("CBL" in steps_list_obs):
        print('Computing Obs CBLs')
        # Read in the list of CBL files
        cbl_nseasons = int(os.environ['CBL_NUM_SEASONS'])
        with open(obs_cbl_filetxt) as ocl:
            obs_infiles = ocl.read().splitlines()
        if (obs_infiles[0] == 'file_list'):
            obs_infiles = obs_infiles[1:]
        if len(obs_infiles) != (cbl_nseasons*dseasons):
            raise Exception('Invalid Obs data; each year must contain the same date range to␣
↪calculate seasonal averages.')
        cbls_obs,lats_obs,lons_obs,mhweight_obs,cbl_time_obs = steps_obs.run_CBL(obs_infiles,
```

(continues on next page)

(continued from previous page)

```
↪cbl_nseasons,dseasons)

   if ("CBL" in steps_list_fcst) and (use_cbl_obs == 'false'):
       # Add in step to use obs for CBLS
       print('Computing Forecast CBLs')
       cbl_nseasons = int(os.environ['CBL_NUM_SEASONS'])
       with open(fcst_cbl_filetxt) as fcl:
           fcst_infiles = fcl.read().splitlines()
       if (fcst_infiles[0] == 'file_list'):
           fcst_infiles = fcst_infiles[1:]
       if len(fcst_infiles) != (cbl_nseasons*dseasons):
           raise Exception('Invalid Fcst data; each year must contain the same date range␣
↪to calculate seasonal averages.')
       cbls_fcst,lats_fcst,lons_fcst,mhweight_fcst,cbl_time_fcst = steps_fcst.run_CBL(fcst_
↪infiles,cbl_nseasons,dseasons)
   elif ("CBL" in steps_list_fcst) and (use_cbl_obs == 'true'):
       if not ("CBL" in steps_list_obs):
           raise Exception('Must run observed CBLs before using them as a forecast.')
       cbls_fcst = cbls_obs
       lats_fcst = lats_obs
       lons_fcst = lons_obs
       mhweight_fcst = mhweight_obs
       cbl_time_fcst = cbl_time_obs

   #Plot Central Blocking Latitude
   if ("PLOTCBL" in steps_list_obs):
       if not ("CBL" in steps_list_obs):
           raise Exception('Must run observed CBLs before plotting them.')
       print('Plotting Obs CBLs')
       cbl_plot_mthstr = os.environ['OBS_CBL_PLOT_MTHSTR']
       cbl_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_CBL_PLOT_OUTPUT_NAME',
↪'obs_cbl_avg'))
       create_cbl_plot(lons_obs, lats_obs, cbls_obs, mhweight_obs, cbl_plot_mthstr, cbl_
↪plot_outname,
           do_averaging=True)
   if ("PLOTCBL" in steps_list_fcst):
       if not ("CBL" in steps_list_fcst):
           raise Exception('Must run forecast CBLs before plotting them.')
       print('Plotting Forecast CBLs')
       cbl_plot_mthstr = os.environ['FCST_CBL_PLOT_MTHSTR']
       cbl_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_CBL_PLOT_OUTPUT_NAME',
↪'fcst_cbl_avg'))
       create_cbl_plot(lons_fcst, lats_fcst, cbls_fcst, mhweight_fcst, cbl_plot_mthstr, cbl_
↪plot_outname,
           do_averaging=True)
```

(continues on next page)

(continued from previous page)

```
    # Run IBL
    if ("IBL" in steps_list_obs):
        if not ("CBL" in steps_list_obs):
            raise Exception('Must run observed CBLs before running IBLs.')
        print('Computing Obs IBLs')
        ibl_nseasons = int(os.environ['IBL_NUM_SEASONS'])
        with open(obs_ibl_filetxt) as oil:
            obs_infiles = oil.read().splitlines()
        if (obs_infiles[0] == 'file_list'):
            obs_infiles = obs_infiles[1:]
        if len(obs_infiles) != (ibl_nseasons*dseasons):
            raise Exception('Invalid Obs data; each year must contain the same date range to␣
↪calculate seasonal averages.')
        ibls_obs,ibl_time_obs = steps_obs.run_Calc_IBL(cbls_obs,obs_infiles,ibl_nseasons,
↪dseasons)
        daynum_obs = np.arange(0,len(ibls_obs[0,:,0]),1)
    if ("IBL" in steps_list_fcst):
        if (not "CBL" in steps_list_fcst):
            raise Exception('Must run forecast CBLs or use observed CBLs before running IBLs.
↪')
        print('Computing Forecast IBLs')
        ibl_nseasons = int(os.environ['IBL_NUM_SEASONS'])
        with open(fcst_ibl_filetxt) as fil:
            fcst_infiles = fil.read().splitlines()
        if (fcst_infiles[0] == 'file_list'):
            fcst_infiles = fcst_infiles[1:]
        if len(fcst_infiles) != (ibl_nseasons*dseasons):
            raise Exception('Invalid Fcst data; each year must contain the same date range␣
↪to calculate seasonal averages.')
        ibls_fcst,ibl_time_fcst = steps_fcst.run_Calc_IBL(cbls_fcst,fcst_infiles,ibl_
↪nseasons,dseasons)
        daynum_fcst = np.arange(0,len(ibls_fcst[0,:,0]),1)

    if ("IBL" in steps_list_obs) and ("IBL" in steps_list_fcst):
        # Print IBLs to output matched pair file
        i_mpr_outdir = os.path.join(mpr_dir,'IBL')
        if not os.path.exists(i_mpr_outdir):
            os.makedirs(i_mpr_outdir)
        modname = os.environ.get('MODEL_NAME','GFS')
        maskname = os.environ.get('MASK_NAME','FULL')
        ibl_outfile_prefix = os.path.join(i_mpr_outdir,'IBL_stat_'+modname)
        cbls_avg = np.nanmean(cbls_obs,axis=0)
        write_mpr_file(ibls_obs,ibls_fcst,cbls_avg,lons_obs,ibl_time_obs,ibl_time_fcst,
```

(continues on next page)

(continued from previous page)

```
↪modname,
           'NA','IBLs','block','Z500','IBLs','block','Z500',maskname,'500',ibl_outfile_
↪prefix)

   # Plot IBLS
   if("PLOTIBL" in steps_list_obs) and not ("PLOTIBL" in steps_list_fcst):
       if not ("IBL" in steps_list_obs):
           raise Exception('Must run observed IBLs before plotting them.')
       print('Plotting Obs IBLs')
       ibl_plot_title = os.environ.get('OBS_IBL_PLOT_TITLE','Instantaneous Blocked Longitude
↪')
       ibl_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_IBL_PLOT_OUTPUT_NAME',
↪'obs_IBL_Freq'))
       ibl_plot_label1 = os.environ.get('IBL_PLOT_OBS_LABEL','')
       pb.plot_ibls(ibls_obs,lons_obs,ibl_plot_title,ibl_plot_outname,label1=ibl_plot_
↪label1)
   elif ("PLOTIBL" in steps_list_fcst) and not ("PLOTIBL" in steps_list_obs):
       if not ("IBL" in steps_list_fcst):
           raise Exception('Must run forecast IBLs before plotting them.')
       print('Plotting Forecast IBLs')
       ibl_plot_title = os.environ.get('FCST_IBL_PLOT_TITLE','Instantaneous Blocked␣
↪Longitude')
       ibl_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_IBL_PLOT_OUTPUT_NAME',
↪'fcst_IBL_Freq'))
       ibl_plot_label1 = os.environ.get('IBL_PLOT_FCST_LABEL','')
       pb.plot_ibls(ibls_fcst,lons_fcst,ibl_plot_title,ibl_plot_outname,label1=ibl_plot_
↪label1)
   elif ("PLOTIBL" in steps_list_obs) and ("PLOTIBL" in steps_list_fcst):
       if (not "IBL" in steps_list_obs) and (not "IBL" in steps_list_fcst):
           raise Exception('Must run forecast and observed IBLs before plotting them.')
       print('Plotting Obs and Forecast IBLs')
       ibl_plot_title = os.environ['IBL_PLOT_TITLE']
       ibl_plot_outname = os.path.join(oplot_dir,os.environ.get('IBL_PLOT_OUTPUT_NAME','IBL_
↪Freq'))
       #Check to see if there are plot legend labels
       ibl_plot_label1 = os.environ.get('IBL_PLOT_OBS_LABEL','Observation')
       ibl_plot_label2 = os.environ.get('IBL_PLOT_FCST_LABEL','Forecast')
       pb.plot_ibls(ibls_obs,lons_obs,ibl_plot_title,ibl_plot_outname,data2=ibls_fcst,
↪lon2=lons_fcst,
           label1=ibl_plot_label1,label2=ibl_plot_label2)


   # Run GIBL
   if ("GIBL" in steps_list_obs):
       if not ("IBL" in steps_list_obs):
```

(continues on next page)

(continued from previous page)

```
            raise Exception('Must run observed IBLs before running GIBLs.')
        print('Computing Obs GIBLs')
        gibls_obs = steps_obs.run_Calc_GIBL(ibls_obs,lons_obs)

    if ("GIBL" in steps_list_fcst):
        if not ("IBL" in steps_list_fcst):
            raise Exception('Must run Forecast IBLs before running GIBLs.')
        print('Computing Forecast GIBLs')
        gibls_fcst = steps_fcst.run_Calc_GIBL(ibls_fcst,lons_fcst)


    # Calc Blocks
    if ("CALCBLOCKS" in steps_list_obs):
        if not ("GIBL" in steps_list_obs):
            raise Exception('Must run observed GIBLs before calculating blocks.')
        print('Computing Obs Blocks')
        block_freq_obs = steps_obs.run_Calc_Blocks(ibls_obs,gibls_obs,lons_obs,daynum_obs)

    if ("CALCBLOCKS" in steps_list_fcst):
        if not ("GIBL" in steps_list_fcst):
            raise Exception('Must run Forecast GIBLs before calculating blocks.')
        print('Computing Forecast Blocks')
        block_freq_fcst = steps_fcst.run_Calc_Blocks(ibls_fcst,gibls_fcst,lons_fcst,daynum_
↪fcst)

    # Write out a Blocking MPR file if both obs and forecast blocking calculation performed
    if ("CALCBLOCKS" in steps_list_obs) and ("CALCBLOCKS" in steps_list_fcst):
        b_mpr_outdir = os.path.join(mpr_dir,'Blocks')
        if not os.path.exists(b_mpr_outdir):
            os.makedirs(b_mpr_outdir)
        # Print Blocks to output matched pair file
        modname = os.environ.get('MODEL_NAME','GFS')
        maskname = os.environ.get('MASK_NAME','FULL')
        blocks_outfile_prefix = os.path.join(b_mpr_outdir,'blocking_stat_'+modname)
        cbls_avg = np.nanmean(cbls_obs,axis=0)
        write_mpr_file(block_freq_obs,block_freq_fcst,cbls_avg,lons_obs,ibl_time_obs,ibl_
↪time_fcst,modname,
            'NA','Blocks','block','Z500','Blocks','block','Z500',maskname,'500',blocks_
↪outfile_prefix)


    # Plot Blocking Frequency
    if ("PLOTBLOCKS" in steps_list_obs):
        if not ("CALCBLOCKS" in steps_list_obs):
            raise Exception('Must compute observed blocks before plotting them.')
```

(continues on next page)

(continued from previous page)

```
        print('Plotting Obs Blocks')
        blocking_plot_title = os.environ.get('OBS_BLOCKING_PLOT_TITLE','Obs Blocking␣
↪Frequency')
        blocking_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_BLOCKING_PLOT_
↪OUTPUT_NAME','obs_Block_Freq'))
        pb.plot_blocks(block_freq_obs,gibls_obs,ibls_obs,lons_obs,blocking_plot_title,
↪blocking_plot_outname)
    if ("PLOTBLOCKS" in steps_list_fcst):
        if not ("CALCBLOCKS" in steps_list_fcst):
            raise Exception('Must compute forecast blocks before plotting them.')
        print('Plotting Forecast Blocks')
        blocking_plot_title = os.environ.get('FCST_BLOCKING_PLOT_TITLE','Forecast Blocking␣
↪Frequency')
        blocking_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_BLOCKING_PLOT_
↪OUTPUT_NAME','fcst_Block_Freq'))
        pb.plot_blocks(block_freq_fcst,gibls_fcst,ibls_fcst,lons_fcst,blocking_plot_title,
↪blocking_plot_outname)


if __name__ == "__main__":
    main()
```

Running METplus

This use case is run in the following ways:

1) Passing in UserScript_fcstGFS_obsERA_Blocking.py then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_fcstGFS_obsERA_Blocking.py -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_fcstGFS_obsERA_Blocking.py:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_fcstGFS_obsERA_Blocking.py
```

The following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/s2s_mid_lat/Blocking (relative to **OUTPUT_BASE**) and will contain output for the steps requested. This may include the regridded data, daily averaged files, running mean files, and anomaly files. In addition, output CBL, IBL, and Blocking frequency plots can be generated. The location of these output plots can be specified as BLOCKING_PLOT_OUTPUT_DIR. If it is not specified, plots will be sent to OUTPUT_BASE/plots. MET format matched pair output will also be generated for IBLs and blocks if a user runs these steps on both the model and observation data. The location the matched pair output can be specified as BLOCKING_MPR_OUTPUT_DIR. If it is not specified, plots will be sent to OUTPUT_BASE/mpr. An output contingency table statistics line from stat_analysis is also generated from the IBL and Blocks matched pair files. The location of the output is set as STAT_ANALYSIS_OUTPUT_DIR.

Keywords

Note:

- RegridDataPlaneToolUseCase
- PCPCombineToolUseCase
- StatAnalysisToolUseCase
- S2SAppUseCase
- S2SMidLatAppUseCase
- NetCDFFileUseCase
- GRIB2FileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s_mid_lat-UserScript_fcstGFS_obsERA_Blocking.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.11.4 WeatherRegime Calculation: GFS and ERA RegridDataPlane, PcpCombine, and WeatherRegime python code

model_applications/ s2s_mid_lat/ UserScript_fcstGFS_obsERA_WeatherRegime.py

Scientific Objective

To perform a weather regime analysis using 500 mb height data. There are 2 pre- processing steps, RegridDataPlane and PcpCombine, and 4 steps in the weather regime analysis, elbow, EOFs, K means, and the Time frequency. The elbow and K means steps begin with K means clustering. Elbow then computes the sum of squared distances for clusters 1 - 14 and draws a straight line from the sum of squared distance for the clusters. This helps determine the optimal cluster number by examining the largest difference between the curve and the straight line. The EOFs step is optional. It computes an empirical orthogonal function analysis. The K means step uses clustering to compute the frequency of occurrence and anomalies for each cluster to give the most common weather regimes. Then, the time frequency computes the frequency of each weather regime over a user specified time frame. Finally, stat_analysis can be run to compute an categorical analysis of the weather regime classification or an anomaly correlation of the time frequency data.

Datasets

- Forecast dataset: GFS Forecast 500 mb height.
- Observation dataset: ERA Reanlaysis 500 mb height.

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* numpy
* netCDF4
* datetime
* pylab
* scipy
* sklearn
* eofs
```

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case runs the weather regime driver script which runs the steps the user lists in STEPS_OBS. The possible steps are regridding, time averaging, creating a list of input files for the weather regime calculation, computing the elbow (ELBOW), plotting the elbow (PLOTELBOW), computing EOFs (EOF), plotting EOFs (PLOTEOF), computing K means (KMEANS), plotting the K means (PLOTKMEANS), computing a time frequency of weather regimes (TIMEFREQ) and plotting the time frequency (PLOTFREQ). All variables are set up in the UserScript .conf file. The pre- processing steps and stat_analysis are listed in the process list, and are formatted as follows:

PROCESS_LIST = RegridDataPlane(regrid_obs), PcpCombine(daily_mean_obs), UserScript(script_wr)

The other steps are listed in the [user_env_vars] section of the UserScript .conf file in the following format: OBS_STEPS = ELBOW+PLOTELBOW+EOF+PLOTEOF+KMEANS+PLOTKMEANS+TIMEFREQ+PLOTFREQ FCST_STEPS = ELBOW+PLOTELBOW+EOF+PLOTEOF+KMEANS+PLOTKMEANS+TIMEFREQ+PLOTFREQ

METplus Workflow

The weather regime python code is run for each time for the forecast and observations data. This example loops by valid time. This version is set to only process the weather regime steps (ELBOW, PLOTELBOW, EOF, PLOTEOF, KMEANS, PLOTKMEANS, TIMEFREQ, PLOTFREQ) and stat_analysis, omitting the regridding, time averaging, and creating the file list pre-processing steps. However, the configurations for pre-processing are available for user reference.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/s2s_mid_lat/UserScript_fcstGFS_obsERA_WeatherRegime.py. The file UserScript_fcstGFS_obsERA_WeatherRegime.conf runs the python program and sets the variables for all steps of the Weather Regime use case including data paths.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s_mid_lat/
↪UserScript_fcstGFS_obsERA_WeatherRegime.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###
```

(continues on next page)

(continued from previous page)

```
# All steps, including pre-processing:
# PROCESS_LIST = RegridDataPlane(regrid_obs), PcpCombine(daily_mean_obs), UserScript(script_
↪wr), StatAnalysis(sanal_wrclass), StatAnalysis(sanal_wrfreq)
# Weather Regime Analysis and stat_analysis:

PROCESS_LIST = UserScript(script_wr), StatAnalysis(sanal_wrclass), StatAnalysis(sanal_wrfreq)


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
#VALID_BEG = 1979120100
VALID_BEG = 2000120100
VALID_END = 2017022800
VALID_INCREMENT = 86400

LEAD_SEQ = 0

# Only Process DJF
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:1201,0229"


###
# RegridDataPlane(regrid_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Regridding Pre-Processing Step
[regrid_obs]

VALID_BEG = 2000120200
VALID_END = 2017022818
VALID_INCREMENT = 21600
```

(continues on next page)

(continued from previous page)

```
# REGRID_DATA_PLANE (Pre Processing Step 1), currently turned off
# Run regrid_data_plane on forecast data
OBS_REGRID_DATA_PLANE_RUN = True

OBS_DATA_PLANE_ONCE_PER_FIELD = False

OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Z
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = P500
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = Z500

# Mask to use for regridding
# A 1 degree latitude/longitude grid running 24 to 54 degrees latitude
# and 230 to 300 degrees longitude
REGRID_DATA_PLANE_VERIF_GRID = latlon 71 31 54 230 -1.0 1.0

# Method to run regrid_data_plane, not setting this will default to NEAREST
REGRID_DATA_PLANE_METHOD = BILIN

# Regridding width used in regrid_data_plane, not setting this will default to 1
REGRID_DATA_PLANE_WIDTH = 2

OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mid_lat/UserScript_
↪fcstGFS_obsERA_WeatherRegime/ERA/OrigData
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪WeatherRegime/ERA/Regrid

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = {valid?fmt=%Y%m}/ei.oper.an.pl.regn128sc.{valid?fmt=%Y
↪%m%d%H}
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/Z500_6hourly_{init?fmt=%Y%m%d%H}_
↪NH.nc


###
# PCPCombine(daily_mean_obs) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Daily Mean Pre-Processing Step
[daily_mean_obs]

VALID_BEG = 2000120218
VALID_END = 2017022818

OBS_PCP_COMBINE_RUN = True
```

(continues on next page)

(continued from previous page)

```
OBS_PCP_COMBINE_METHOD = DERIVE
OBS_PCP_COMBINE_STAT_LIST = MEAN

OBS_PCP_COMBINE_INPUT_ACCUMS = 6
OBS_PCP_COMBINE_INPUT_NAMES = Z500
OBS_PCP_COMBINE_INPUT_LEVELS = "(*,*)"
OBS_PCP_COMBINE_INPUT_OPTIONS = convert(x) = x / 9.81; set_attr_valid = "{valid?fmt=%Y%m%d_%H
↪%M%S?shift=-64800}";

# Convert height and derive mean over 24 hours
OBS_PCP_COMBINE_OUTPUT_ACCUM = 24
OBS_PCP_COMBINE_DERIVE_LOOKBACK = 24

# Name output variable Z500
OBS_PCP_COMBINE_OUTPUT_NAME = Z500

OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪WeatherRegime/ERA/Regrid
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪WeatherRegime/ERA/Daily

OBS_PCP_COMBINE_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/Z500_6hourly_{valid?fmt=%Y%m%d%H}_NH.nc
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = Z500_daily_{valid?fmt=%Y%m%d?shift=-64800}_NH.nc


# Variables for the Weather Regime code
[user_env_vars]
# Steps to Run
FCST_STEPS = ELBOW+PLOTELBOW+EOF+PLOTEOF+KMEANS+PLOTKMEANS+TIMEFREQ+PLOTFREQ
OBS_STEPS = ELBOW+PLOTELBOW+EOF+PLOTEOF+KMEANS+PLOTKMEANS+TIMEFREQ+PLOTFREQ

# Make OUTPUT_BASE Available to the script
SCRIPT_OUTPUT_BASE = {OUTPUT_BASE}

# Number of Seasons and Days per season that should be available
# The code will fill missing data, but requires the same number of days per
# season for each year.  You may need to omit leap days if February is part of
# the processing
NUM_SEASONS = 17
DAYS_PER_SEASON = 89

# Variable for the Z500 data
OBS_WR_VAR = Z500
```

(continues on next page)

(continued from previous page)

```
FCST_WR_VAR = Z500_P500

# Weather Regime Number
OBS_WR_NUMBER = 6
FCST_WR_NUMBER = {OBS_WR_NUMBER}

# Number of clusters
OBS_NUM_CLUSTERS = 20
FCST_NUM_CLUSTERS = {OBS_NUM_CLUSTERS}

# Number of principal components
OBS_NUM_PCS = 10
FCST_NUM_PCS = {OBS_NUM_PCS}

# Time (in timesteps) over which to compute weather regime frequencies
# i.e. if your data time step is days and you want to average over 7
# days, input 7
# Optional, only needed if you want to compute frequencies
OBS_WR_FREQ = 7
FCST_WR_FREQ = {OBS_WR_FREQ}

# These variables control reordering the forecast weather regime to match the
# observations if their orders are different
# REORDER_FCST_MANUAL will use the order in FCST_ORDER, whereas REORDER_FCST will
# use a pattern correlation to reorder
# It is recommended to set REORDER_FCST_MANUAL to False if this is the first time running the
# case
REORDER_FCST = True
REORDER_FCST_MANUAL = False
#Order to use if REORDER_FCST_MANUAL = True; will be ignored if REORER_FCST_MANUAL = False
FCST_ORDER = 1,3,4,2,5,6

# Type, name and directory of Output File for weather regime classification
# Type options are text or netcdf
OBS_WR_OUTPUT_FILE_TYPE = text
OBS_WR_OUTPUT_FILE = obs_weather_regime_class
FCST_WR_OUTPUT_FILE_TYPE = text
FCST_WR_OUTPUT_FILE = fcst_weather_regime_class
WR_OUTPUT_FILE_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_WeatherRegime

# Directory to send output plots
WR_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_WeatherRegime/plots/

# Elbow Plot Title and output file name
OBS_ELBOW_PLOT_TITLE = ERA Elbow Method For Optimal k
```

(continues on next page)

(continued from previous page)

```
OBS_ELBOW_PLOT_OUTPUT_NAME = obs_elbow
FCST_ELBOW_PLOT_TITLE = GFS Elbow Method For Optimal k
FCST_ELBOW_PLOT_OUTPUT_NAME = fcst_elbow


# EOF plot output name and contour levels
OBS_EOF_PLOT_OUTPUT_NAME = obs_eof
FCST_EOF_PLOT_OUTPUT_NAME = fcst_eof
EOF_PLOT_LEVELS = -50, -45, -40, -35, -30, -25, -20, -15, -10, -5,  0,  5, 10, 15, 20, 25,␣
↪30, 35, 40, 45, 50

# K means Plot Output Name and contour levels
OBS_KMEANS_PLOT_OUTPUT_NAME = obs_kmeans
FCST_KMEANS_PLOT_OUTPUT_NAME = fcst_kmeans
KMEANS_PLOT_LEVELS = -80, -70, -60, -50, -40, -30, -20, -10, 0, 10, 20, 30, 40, 50, 60, 70,␣
↪80

# Frequency Plot title and output file name
OBS_FREQ_PLOT_TITLE = ERA Seasonal Cycle of WR Days/Week (2000-2017)
OBS_FREQ_PLOT_OUTPUT_NAME = obs_freq
FCST_FREQ_PLOT_TITLE = GFS Seasonal Cycle of WR Days/Week (2000-2017)
FCST_FREQ_PLOT_OUTPUT_NAME = fcst_freq

# MPR file information
MASK_NAME = FULL
WR_MPR_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_WeatherRegime/mpr


###
# UserScript(script_wr) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Run the Weather Regime Script
[script_wr]

LEAD_SEQ = 24

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

# Template of filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mid_lat/UserScript_fcstGFS_
↪obsERA_WeatherRegime/ERA/Daily/Z500_daily_{valid?fmt=%Y%m%d}_NH.nc,{INPUT_BASE}/model_
↪applications/s2s_mid_lat/UserScript_fcstGFS_obsERA_WeatherRegime/GFS/Daily/Z500_{init?fmt=
↪%Y%m%d}_{lead?fmt=%HHH}_NH.nc
```

(continues on next page)

(continued from previous page)

```
# Name of the file containing the listing of input files
# The options are OBS_INPUT for observations or FCST_INPUT for forecast
# Or, set OBS_INPUT, FCST_INPUT if doing both and make sure the USER_SCRIPT_INPUT_TEMPLATE_
↪is ordered:
# observation_template, forecast_template
USER_SCRIPT_INPUT_TEMPLATE_LABELS = OBS_INPUT, FCST_INPUT

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_fcstGFS_obsERA_WeatherRegime/WeatherRegime_driver.py


###
# StatAnalysis(sanal_wrclass) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#statanalysis
###

[sanal_wrclass]

VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20001202
VALID_END = 20170228

STAT_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/STATAnalysisConfig_wrapped

MODEL1 = GFS
MODEL1_OBTYPE = ADPUPA

STAT_ANALYSIS_JOB_NAME = aggregate_stat
STAT_ANALYSIS_JOB_ARGS = -out_line_type MCTS -out_thresh >=1,>=2,>=3,>=4,>=5 -out_stat [out_
↪stat_file]

MODEL_LIST = {MODEL1}
FCST_LEAD_LIST = 24
LINE_TYPE_LIST = MPR

GROUP_LIST_ITEMS = MODEL_LIST
LOOP_LIST_ITEMS = FCST_LEAD_LIST

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪WeatherRegime/mpr/WeatherRegime

STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_WeatherRegime
```

(continues on next page)

(continued from previous page)

```
MODEL1_STAT_ANALYSIS_OUT_STAT_TEMPLATE = {model?fmt=%s}_ERA_WRClass_{lead?fmt=%H%M%S}L_MCTS.
↪stat


###
# StatAnalysis(sanal_wrfreq) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#statanalysis
###

[sanal_wrfreq]

VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20001202
VALID_END = 20170228

STAT_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/STATAnalysisConfig_wrapped

MODEL1 = GFS
MODEL1_OBTYPE = ADPUPA

STAT_ANALYSIS_JOB_NAME = aggregate_stat
STAT_ANALYSIS_JOB_ARGS = -out_line_type CNT -by DESC -out_stat [out_stat_file]

MODEL_LIST = {MODEL1}
FCST_LEAD_LIST = 24
LINE_TYPE_LIST = MPR

GROUP_LIST_ITEMS = MODEL_LIST
LOOP_LIST_ITEMS = FCST_LEAD_LIST

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_
↪WeatherRegime/mpr/freq

STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mid_lat/UserScript_fcstGFS_obsERA_WeatherRegime

MODEL1_STAT_ANALYSIS_OUT_STAT_TEMPLATE = {model?fmt=%s}_ERA_WR_freq_{lead?fmt=%H%M%S}L_CNT.
↪stat
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

See the following files for more information about the environment variables set in this configuration file.

parm/use_cases/met_tool_wrapper/RegridDataPlane/RegridDataPlane.py parm/use_cases/met_tool_wrapper/PCPCo
parm/use_cases/met_tool_wrapper/StatAnalysis/StatAnalysis.py

Python Scripts

This use case uses Python scripts to perform the blocking calculation

parm/use_cases/model_applications/s2s_mid_lat/UserScript_fcstGFS_obsERA_WeatherRegime/WeatherRegime_drive
This script calls the requested steps in the blocking analysis for a forecast, observation, or both. The possible steps are computing the elbow, computing EOFs, and computing weather regimes using k means clustering.

metcalcpy/contributed/blocking_weather_regime/WeatherRegime.py: This script runs the requested steps, containing the code for computing the bend in the elbow, computing EOFs, and computing weather regimes using k means clustering. See the METcalcpy Weather Regime Calculation Script for more information.

metcalcpy/contributed/blocking_weather_regime/Blocking_WeatherRegime_util.py: This script contains functions used by both the blocking anwd weather regime analysis, including the code for determining which steps the user wants to run, and finding and reading the input files in the format from the output pre-processing steps. See the METcalcpy Utility script for more information.

```
#!/usr/bin/env python3
import sys
import os
import numpy as np
import netCDF4
import warnings

from metcalcpy.contributed.blocking_weather_regime.WeatherRegime import␣
↪WeatherRegimeCalculation
from metcalcpy.contributed.blocking_weather_regime.Blocking_WeatherRegime_util import parse_
↪steps, read_nc_met, write_mpr_file, reorder_fcst_regimes,reorder_fcst_regimes_correlate
from metplotpy.contributed.weather_regime import plot_weather_regime as pwr


def main():
```

(continues on next page)

(continued from previous page)

```
    steps_list_fcst,steps_list_obs = parse_steps()

    if not steps_list_obs and not steps_list_fcst:
        warnings.warn('No processing steps requested for either the model or observations,')
        warnings.warn(' nothing will be run')
        warnings.warn('Set FCST_STEPS and/or OBS_STEPS in the [user_env_vars] section to␣
↪process data')


    ######################################################################
    # Blocking Calculation and Plotting
    ######################################################################
    # Set up the data
    steps_obs = WeatherRegimeCalculation('OBS')
    steps_fcst = WeatherRegimeCalculation('FCST')

    # Check to see if there is a plot directory
    oplot_dir = os.environ.get('WR_PLOT_OUTPUT_DIR','')
    obase = os.environ['SCRIPT_OUTPUT_BASE']
    if not oplot_dir:
        oplot_dir = os.path.join(obase,'plots')
    if not os.path.exists(oplot_dir):
        os.makedirs(oplot_dir)

     # Check to see if there is a mpr output directory
    mpr_outdir = os.environ.get('WR_MPR_OUTPUT_DIR','')
    if not mpr_outdir:
        mpr_outdir = os.path.join(obase,'mpr')

     # Get number of seasons and days per season
    nseasons = int(os.environ['NUM_SEASONS'])
    dseasons = int(os.environ['DAYS_PER_SEASON'])

    # Grab the Daily text files
    obs_wr_filetxt = os.environ.get('METPLUS_FILELIST_OBS_INPUT','')
    fcst_wr_filetxt = os.environ.get('METPLUS_FILELIST_FCST_INPUT','')

    if ("ELBOW" in steps_list_obs) or ("EOF" in steps_list_obs) or ("KMEANS" in steps_list_
↪obs):
        with open(obs_wr_filetxt) as owl:
            obs_infiles = owl.read().splitlines()
        # Remove the first line if it's there
        if (obs_infiles[0] == 'file_list'):
            obs_infiles = obs_infiles[1:]
        if len(obs_infiles) != (nseasons*dseasons):
```

(continues on next page)

(continued from previous page)

```
            raise Exception('Invalid Obs data; each year must contain the same date range to␣
↪calculate seasonal averages.')
        obs_invar = os.environ.get('OBS_WR_VAR','')
        z500_obs,lats_obs,lons_obs,timedict_obs = read_nc_met(obs_infiles,obs_invar,nseasons,
↪dseasons)
        z500_detrend_obs,z500_detrend_2d_obs = steps_obs.weights_detrend(lats_obs,lons_obs,
↪z500_obs)

    if ("ELBOW" in steps_list_fcst) or ("EOF" in steps_list_fcst) or("KMEANS" in steps_list_
↪fcst):
        with open(fcst_wr_filetxt) as fwl:
            fcst_infiles = fwl.read().splitlines()
        # Remove the first line if it's there
        if (fcst_infiles[0] == 'file_list'):
            fcst_infiles = fcst_infiles[1:]
        if len(fcst_infiles) != (nseasons*dseasons):
            raise Exception('Invalid Obs data; each year must contain the same date range to␣
↪calculate seasonal averages.')
        fcst_invar = os.environ.get('FCST_WR_VAR','')
        z500_fcst,lats_fcst,lons_fcst,timedict_fcst = read_nc_met(fcst_infiles,fcst_invar,
↪nseasons,dseasons)
        z500_detrend_fcst,z500_detrend_2d_fcst = steps_fcst.weights_detrend(lats_fcst,lons_
↪fcst,z500_fcst)


    if ("ELBOW" in steps_list_obs):
        print('Running Obs Elbow')
        K_obs,d_obs,mi_obs,line_obs,curve_obs = steps_obs.run_elbow(z500_detrend_2d_obs)

    if ("ELBOW" in steps_list_fcst):
        print('Running Forecast Elbow')
        K_fcst,d_fcst,mi_fcst,line_fcst,curve_fcst = steps_fcst.run_elbow(z500_detrend_2d_
↪fcst)

    if ("PLOTELBOW" in steps_list_obs):
        if not ("ELBOW" in steps_list_obs):
            raise Exception('Must run observed Elbow before plotting observed elbow.')
        print('Creating Obs Elbow plot')
        elbow_plot_title = os.environ.get('OBS_ELBOW_PLOT_TITLE','Elbow Method For Optimal k
↪')
        elbow_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_ELBOW_PLOT_OUTPUT_
↪NAME','obs_elbow'))
        pwr.plot_elbow(K_obs,d_obs,mi_obs,line_obs,curve_obs,elbow_plot_title,elbow_plot_
↪outname)
```

(continues on next page)

(continued from previous page)

```
    if ("PLOTELBOW" in steps_list_fcst):
        if not ("ELBOW" in steps_list_fcst):
            raise Exception('Must run forecast Elbow before plotting forecast elbow.')
        print('Creating Forecast Elbow plot')
        elbow_plot_title = os.environ.get('FCST_ELBOW_PLOT_TITLE','Elbow Method For Optimal k
↪')
        elbow_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_ELBOW_PLOT_OUTPUT_
↪NAME','fcst_elbow'))
        pwr.plot_elbow(K_fcst,d_fcst,mi_fcst,line_fcst,curve_fcst,elbow_plot_title,elbow_
↪plot_outname)


    if ("EOF" in steps_list_obs):
        print('Running Obs EOF')
        eof_obs,pc_obs,wrnum_obs,variance_fractions_obs = steps_obs.Calc_EOF(z500_obs)
        z500_detrend_2d_obs = steps_obs.reconstruct_heights(eof_obs,pc_obs,z500_detrend_2d_
↪obs.shape)

    if ("EOF" in steps_list_fcst):
        print('Running Forecast EOF')
        eof_fcst,pc_fcst,wrnum_fcst,variance_fractions_fcst = steps_fcst.Calc_EOF(z500_fcst)
        z500_detrend_2d_fcst = steps_fcst.reconstruct_heights(eof_fcst,pc_fcst,z500_detrend_
↪2d_fcst.shape)

    if ("PLOTEOF" in steps_list_obs):
        if not ("EOF" in steps_list_obs):
            raise Exception('Must run observed EOFs before plotting observed EOFs.')
        print('Plotting Obs EOFs')
        pltlvls_str = os.environ['EOF_PLOT_LEVELS'].split(',')
        pltlvls = [float(pp) for pp in pltlvls_str]
        eof_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_EOF_PLOT_OUTPUT_NAME',
↪'obs_eof'))
        pwr.plot_eof(eof_obs,wrnum_obs,variance_fractions_obs,lons_obs,lats_obs,eof_plot_
↪outname,pltlvls)

    if ("PLOTEOF" in steps_list_fcst):
        if not ("EOF" in steps_list_fcst):
            raise Exception('Must run forecast EOFs before plotting forecast EOFs.')
        print('Plotting Forecast EOFs')
        pltlvls_str = os.environ['EOF_PLOT_LEVELS'].split(',')
        pltlvls = [float(pp) for pp in pltlvls_str]
        eof_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_EOF_PLOT_OUTPUT_NAME',
↪'fcst_eof'))
        pwr.plot_eof(eof_fcst,wrnum_fcst,variance_fractions_fcst,lons_fcst,lats_fcst,eof_
↪plot_outname,pltlvls)
```

(continues on next page)

(continued from previous page)

```
    if ("KMEANS" in steps_list_obs):
        print('Running Obs K Means')
        kmeans_obs,wrnum_obs,perc_obs,wrc_obs= steps_obs.run_K_means(z500_detrend_2d_obs,
↪timedict_obs,z500_obs.shape)
        steps_obs.write_K_means_file(timedict_obs,wrc_obs)

    if ("KMEANS" in steps_list_fcst):
        print('Running Forecast K Means')
        kmeans_fcst,wrnum_fcst,perc_fcst,wrc_fcst = steps_fcst.run_K_means(z500_detrend_2d_
↪fcst,timedict_fcst,
            z500_fcst.shape)
        reorder_fcst = os.environ.get('REORDER_FCST','False').lower()
        reorder_fcst_manual = os.environ.get('REORDER_FCST_MANUAL','False').lower()
        if (reorder_fcst == 'true') and ("KMEANS" in steps_list_obs):
            kmeans_fcst,perc_fcst,wrc_fcst = reorder_fcst_regimes_correlate(kmeans_obs,
↪kmeans_fcst,perc_fcst,wrc_fcst,wrnum_fcst)
        if reorder_fcst_manual == 'true':
            fcst_order_str = os.environ['FCST_ORDER'].split(',')
            fcst_order = [int(fo) for fo in fcst_order_str]
            kmeans_fcst,perc_fcst,wrc_fcst = reorder_fcst_regimes(kmeans_fcst,perc_fcst,wrc_
↪fcst,wrnum_fcst,fcst_order)
        steps_fcst.write_K_means_file(timedict_fcst,wrc_fcst)


        # Write matched pair output for weather regime classification
        modname = os.environ.get('MODEL_NAME','GFS')
        maskname = os.environ.get('MASK_NAME','FULL')
        mpr_full_outdir = os.path.join(mpr_outdir,'WeatherRegime')
        wr_outfile_prefix = os.path.join(mpr_full_outdir,'weather_regime_stat_'+modname)
        wrc_obs_mpr = wrc_obs[:,:,np.newaxis]
        wrc_fcst_mpr = wrc_fcst[:,:,np.newaxis]
        if not os.path.exists(mpr_full_outdir):
            os.makedirs(mpr_full_outdir)
        write_mpr_file(wrc_obs_mpr,wrc_fcst_mpr,[0.0],[0.0],timedict_obs,timedict_fcst,
↪modname,'NA',
            'WeatherRegimeClass','class','Z500','WeatherRegimeClass','class','Z500',maskname,
↪'500',wr_outfile_prefix)

    if ("PLOTKMEANS" in steps_list_obs):
        if not ("KMEANS" in steps_list_obs):
            raise Exception('Must run observed Kmeans before plotting observed Kmeans.')
        print('Plotting Obs K Means')
        pltlvls_str = os.environ['KMEANS_PLOT_LEVELS'].split(',')
```

(continues on next page)

(continued from previous page)

```
        pltlvls = [float(pp) for pp in pltlvls_str]
        kmeans_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_KMEANS_PLOT_OUTPUT_
↪NAME','obs_kmeans'))
        pwr.plot_K_means(kmeans_obs,wrnum_obs,lons_obs,lats_obs,perc_obs,kmeans_plot_outname,
↪pltlvls)

    if ("PLOTKMEANS" in steps_list_fcst):
        if not ("KMEANS" in steps_list_fcst):
            raise Exception('Must run forecast Kmeans before plotting forecast Kmeans.')
        print('Plotting Forecast K Means')
        pltlvls_str = os.environ['KMEANS_PLOT_LEVELS'].split(',')
        pltlvls = [float(pp) for pp in pltlvls_str]
        kmeans_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_KMEANS_PLOT_OUTPUT_
↪NAME','fcst_kmeans'))
        pwr.plot_K_means(kmeans_fcst,wrnum_fcst,lons_fcst,lats_fcst,perc_fcst,kmeans_plot_
↪outname,pltlvls)


    if ("TIMEFREQ" in steps_list_obs):
        if not ("KMEANS" in steps_list_obs):
            raise Exception('Must run observed Kmeans before running frequencies.')
        wrfreq_obs,dlen_obs,ts_diff_obs = steps_obs.compute_wr_freq(wrc_obs)

    if ("TIMEFREQ" in steps_list_fcst):
        if not ("KMEANS" in steps_list_fcst):
            raise Exception('Must run forecast Kmeans before running frequencies.')
        wrfreq_fcst,dlen_fcst,ts_diff_fcst = steps_fcst.compute_wr_freq(wrc_fcst)

    if ("TIMEFREQ" in steps_list_obs) and ("TIMEFREQ" in steps_list_fcst):
        # Write matched pair output for frequency of each weather regime
        modname = os.environ.get('MODEL_NAME','GFS')
        maskname = os.environ.get('MASK_NAME','FULL')
        mpr_full_outdir = os.path.join(mpr_outdir,'freq')
        timedict_obs_mpr = {'init':timedict_obs['init'][:,ts_diff_obs-1:],
            'valid':timedict_obs['valid'][:,ts_diff_obs-1:],'lead':timedict_obs['lead'][:,ts_
↪diff_obs-1:]}
        timedict_fcst_mpr = {'init':timedict_fcst['init'][:,ts_diff_fcst-1:],
            'valid':timedict_fcst['valid'][:,ts_diff_fcst-1:],'lead':timedict_fcst['lead'][:,
↪ts_diff_fcst-1:]}
        wrfreq_obs_mpr = wrfreq_obs[:,:,:,np.newaxis]
        wrfreq_fcst_mpr = wrfreq_fcst[:,:,:,np.newaxis]
        if not os.path.exists(mpr_full_outdir):
            os.makedirs(mpr_full_outdir)
        for wrn in np.arange(wrnum_obs):
            wr_outfile_prefix = os.path.join(mpr_full_outdir,'weather_regime'+str(wrn+1).
```

(continues on next page)

(continued from previous page)

```
↪zfill(2)+'_freq_stat_'+modname)
            write_mpr_file(wrfreq_obs_mpr[wrn,:,:,:],wrfreq_fcst_mpr[wrn,:,:,:],[0.0],[0.0],
↪timedict_obs,
                timedict_fcst,modname,str(wrn+1).zfill(2),'WeatherRegimeFreq','percent','Z500
↪','WeatherRegimeFreq',
                'percent','Z500',maskname,'500',wr_outfile_prefix)

    if ("PLOTFREQ" in steps_list_obs):
        if not ("TIMEFREQ" in steps_list_obs):
            raise Exception('Must run observed Frequency calculation before plotting the␣
↪frequencies.')
        freq_plot_title = os.environ.get('OBS_FREQ_PLOT_TITLE','Seasonal Cycle of WR Days/
↪Week')
        freq_plot_outname = os.path.join(oplot_dir,os.environ.get('OBS_FREQ_PLOT_OUTPUT_NAME
↪','obs_freq'))
        # Compute mean
        wrmean_obs = np.nanmean(wrfreq_obs,axis=1)
        pwr.plot_wr_frequency(wrmean_obs,wrnum_obs,dlen_obs,freq_plot_title,freq_plot_
↪outname)

    if ("PLOTFREQ" in steps_list_fcst):
        if not ("TIMEFREQ" in steps_list_fcst):
            raise Exception('Must run forecast Frequency calculation before plotting the␣
↪frequencies.')
        freq_plot_title = os.environ.get('FCST_FREQ_PLOT_TITLE','Seasonal Cycle of WR Days/
↪Week')
        freq_plot_outname = os.path.join(oplot_dir,os.environ.get('FCST_FREQ_PLOT_OUTPUT_NAME
↪','fcst_freq'))
        # Compute mean
        wrmean_fcst = np.nanmean(wrfreq_fcst,axis=1)
        pwr.plot_wr_frequency(wrmean_fcst,wrnum_fcst,dlen_fcst,freq_plot_title,freq_plot_
↪outname)


if __name__ == "__main__":
    main()
```

Running METplus

This use case is run in the following ways:

1) Passing in UserScript_fcstGFS_obsERA_WeatherRegime.py then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_fcstGFS_obsERA_WeatherRegime.py -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_fcstGFS_obsERA_WeatherRegime.py:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mid_lat/
↪UserScript_fcstGFS_obsERA_WeatherRegime.py
```

The following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/s2s_mid_lat/WeatherRegime (relative to **OUTPUT_BASE**) and will contain output for the steps requested. This may include the regridded data, daily averaged files, a text file containing the list of input files, and text files for the weather regime classification and time frequency (if KMEANS and TIMEFREQ are run for both the forecast and observation data). In addition, output elbow, EOF, and Kmeans weather regime plots can be generated. The location of these output plots can be specified as WR_OUTPUT_DIR. If it is not specified, plots will be sent to {OUTPUT_BASE}/plots. The output location for the matched pair files can be specified as WR_MPR_OUTPUT_DIR. If it is not specified, it will be sent to {OUTPUT_BASE}/mpr. The output weather regime text or netCDF file location is set in WR_OUTPUT_FILE_DIR. If this is not specified, the output text/netCDF file will be sent to {OUTPUT_BASE}. The stat_analysis contingency table statistics and anomaly correlation files will be sent to the locations given in STAT_ANALYSIS_OUTPUT_DIR for their respective configuration sections.

Keywords

Note:

- RegridDataPlaneToolUseCase
- PCPCombineToolUseCase
- StatAnalysisToolUseCase
- S2SAppUseCase
- S2SMidLatAppUseCase
- NetCDFFileUseCase
- GRIB2FileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s_mid_lat-UserScript_fcstGFS_obsERA_WeatherRegime.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.12 Subseasonal to Seasonal: Madden-Julian Oscillation

Subseasonal-to-Seasonal model configurations relating to the Madden-Julian oscillation

7.2.16.12.1 UserScript: Make a Phase Diagram plot from input RMM or OMI

model_applications/ s2s_mjo/ UserScript_obsERA_obsOnly_PhaseDiagram.py

Scientific Objective

To produce a phase diagram using either OLR based MJO Index (OMI) or the Real-time Multivariate MJO index (RMM)

Datasets

- Forecast dataset: None.
- Observation dataset: ERA Reanlaysis Outgoing Longwave Radiation.

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* numpy
* netCDF4
* datetime
* xarray
* matplotlib
* scipy
* pandas
```

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case runs the Phase Diagram driver which and creates a phase diagram. Inputs to the driver are a text file containing the following columns, yyyy,mm,dd,hh,pc1,pc2,amp for OMI, or yyyy,mm,dd,pc1,pc2,phase,amp,source for RMM.

METplus Workflow

The Phase diagram driver script python code is run for each lead time on the forecast and observations data. This example loops by valid time for the model pre-processing, and valid time for the other steps. It creates the phase diagram plot and a text file listing of the valid times to use in creating the plots.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/s2s_mjo/UserScript_obsERA_obsERA_OMI.conf. The file UserScript_obsERA_obsOnly_PhaseDiagram/PhaseDiagram_driver.py runs the python program and UserScript_obsERA_obsOnly_PhaseDiagram.conf sets the variables for all steps of the use case.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s_mjo/UserScript_
```

(continues on next page)

(continued from previous page)

```
↪obsERA_obsOnly_PhaseDiagram.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript(obs_time_filelist), UserScript(script_PhaseDiagram)


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2012010100
VALID_END = 2012033100
VALID_INCREMENT = 86400

LEAD_SEQ = 0

# variables referenced in other sections

# Run the obs for these cases
OBS_RUN = True
FCST_RUN = False

# Input and Output Directories for the OBS OLR Files and output text file containing the␣
↪file list
OBS_PDTIME_FMT = %Y%m%d-%H%M%S
OBS_PDTIME_INPUT_TEMPLATE = {valid?fmt=%Y%m%d-%H%M%S}
OBS_PDTIME_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_
↪PhaseDiagram/
```

(continues on next page)

(continued from previous page)

```
OBS_PDTIME_OUTPUT_TEMPLATE = time_list_lead{lead?fmt=%HHH}.txt


###
# UserScript(obs_time_filelist) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Create a time file that contains the times we want to filter for plotting
[obs_time_filelist]

# Find the files for each time
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_FOR_EACH

USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_PhaseDiagram/save_input_files_txt.py {OBS_PDTIME_INPUT_TEMPLATE} {OBS_
↪PDTIME_OUTPUT_DIR}/{OBS_PDTIME_OUTPUT_TEMPLATE}


# Configurations for the Phase Diagram Plotting Script
[user_env_vars]
# Whether to Run the model or obs
RUN_OBS = {OBS_RUN}
RUN_FCST = {FCST_RUN}

# Make OUTPUT_BASE Available to the script
SCRIPT_OUTPUT_BASE = {OUTPUT_BASE}

# Index to Plot
PLOT_INDEX = RMM

# Input Directories
OBS_PHASE_DIAGRAM_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_
↪obsOnly_PhaseDiagram

# Input filename template
OBS_PHASE_DIAGRAM_INPUT_FILE = rmm.1x.txt

# Input Time file
OBS_PHASE_DIAGRAM_INPUT_TIMELIST_TEXTFILE = {OBS_PDTIME_OUTPUT_DIR}/{OBS_PDTIME_OUTPUT_
↪TEMPLATE}

OBS_PHASE_DIAGRAM_INPUT_TIME_FMT = {OBS_PDTIME_FMT}

# Plot Output Directory
```

(continues on next page)

(continued from previous page)

```
PHASE_DIAGRAM_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_PhaseDiagram/
↪plots

# Plot Ouptut Name
OBS_PHASE_PLOT_OUTPUT_NAME = RMM_phase_diagram


###
# UserScript(script_PhaseDiagram) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Configurations for UserScript: Run the RMM Analysis driver
[script_PhaseDiagram]
# list of strings to loop over for each run time.
# Run the user script once per lead
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_PhaseDiagram/PhaseDiagram_driver.py
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

Python Scripts

The phase diagram driver script orchestrates the generation of a phase diagram plot: parm/use_cases/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_OMI/PhaseDiagram_driver.py:

```
#!/usr/bin/env python3
"""
Driver Script to read in OMI or RMM indices and plot phase diagram for specified dates.
OMI values can be obtained from https://psl.noaa.gov/mjo/, RMM values can be obtained from
http://www.bom.gov.au/climate/mjo/graphics/rmm.74toRealtime.txt
"""
```

(continues on next page)

(continued from previous page)

```
import os
import atexit
import numpy as np
import pandas as pd
import datetime
import warnings

import metplotpy.contributed.mjo_rmm_omi.plot_mjo_indices as pmi


def handle_exit(obs_timefile,fcst_timefile):
    try:
        os.remove(obs_timefile)
    except:
        pass

    try:
        os.remove(fcst_timefile)
    except:
        pass

def run_phasediagram_steps(inlabel, alldata_timefile, oplot_dir):

    # which index are we plotting
    indexname = os.environ['PLOT_INDEX']

    pltfile = os.path.join(os.environ[inlabel+'_PHASE_DIAGRAM_INPUT_DIR'],
        os.environ[inlabel+'_PHASE_DIAGRAM_INPUT_FILE'])

    # read data from text file
    if indexname=='OMI':
        data = pd.read_csv(pltfile, header=None, delim_whitespace=True, names=['yyyy','mm',
↪'dd','hh','pc1','pc2','amp'],
            parse_dates={'dtime':['yyyy','mm','dd','hh']})
    elif indexname=='RMM':
        data = pd.read_csv(pltfile,  header=None, delim_whitespace=True,
            names=['yyyy','mm','dd', 'pc1','pc2','phase','amp','source'], parse_dates={'dtime
↪':['yyyy','mm','dd']})

    # Get the file with the listing of times and format of this file
    alldata_timefmt = os.environ[inlabel+'_PHASE_DIAGRAM_INPUT_TIME_FMT']

    # Read the file
    with open(alldata_timefile) as at:
```

(continues on next page)

(continued from previous page)

```
        alldata_time = at.read().splitlines()

    keepdata = []
    for dd in alldata_time:
        timeloc = np.where(data.dtime == datetime.datetime.strptime(dd,alldata_timefmt))
        if len(timeloc[0]) > 0:
            for l in timeloc[0]:
                keepdata.append(l)

    pltdata = data.iloc[keepdata]
    dates = np.array(pltdata.dtime.dt.strftime('%Y%m%d').values,dtype=int)
    months = np.array(pltdata.dtime.dt.strftime('%m').values,dtype=int)
    days = np.array(pltdata.dtime.dt.strftime('%d').values,dtype=int)
    PC1 = np.array(pltdata.pc1.values)
    PC2 = np.array(pltdata.pc2.values)

    # plot the phase diagram
    phase_plot_name = os.path.join(oplot_dir,os.environ.get(inlabel+'_PHASE_PLOT_OUTPUT_NAME
↪',inlabel+'_phase'))
    phase_plot_format = os.environ.get(inlabel+'_PHASE_PLOT_OUTPUT_FORMAT','png')

    # plot the phase diagram
    pmi.phase_diagram(indexname,PC1,PC2,dates,months,days,phase_plot_name,'png')


def main():

    obs_timelist = os.path.join(os.environ.get('OBS_PHASE_DIAGRAM_INPUT_DIR',''),
        os.environ.get('OBS_PHASE_DIAGRAM_INPUT_TIMELIST_TEXTFILE',''))
    fcst_timelist = os.path.join(os.environ.get('FCST_PHASE_DIAGRAM_INPUT_DIR',''),
        os.environ.get('FCST_PHASE_DIAGRAM_INPUT_TIMELIST_TEXTFILE',''))
    atexit.register(handle_exit,obs_timelist,fcst_timelist)

    # Check for an output plot directory in the configs.  Create one if it does not exist
    oplot_dir = os.environ.get('PHASE_DIAGRAM_PLOT_OUTPUT_DIR','')
    if not oplot_dir:
        obase = os.environ['OUTPUT_BASE']
        oplot_dir = os.path.join(obase,'plots')
    if not os.path.exists(oplot_dir):
        os.makedirs(oplot_dir)

    # Determine if doing forecast or obs
    run_obs_phasediagram = os.environ.get('RUN_OBS','False').lower()
    run_fcst_phasediagram = os.environ.get('FCST_RUN_FCST','False').lower()
```

(continues on next page)

(continued from previous page)

```
    # Run the steps to compute OMM
    # Observations
    if (run_obs_phasediagram == 'true'):
        run_phasediagram_steps('OBS', obs_timelist, oplot_dir)

    # Forecast
    if (run_fcst_phasediagram == 'true'):
        run_phasediagram_steps('FCST', fcst_timelist, oplot_dir)

    # nothing selected
    if (run_obs_phasediagram == 'false') and (run_fcst_phasediagram == 'false'):
        warnings.warn('Forecast and Obs runs not selected, no plots will be created')
        warnings.warn('Set RUN_FCST or RUN_OBS in the [user_en_vars] section to generate␣
↪output')


if __name__ == "__main__":
    main()
```

```
#! /usr/bin/env python

import os
import sys

input_file = sys.argv[1]
output_file = sys.argv[2]

output_dir = os.path.dirname(output_file)
if not os.path.exists(output_dir):
    print(f'Creating output dir: {output_dir}')
    os.makedirs(output_dir)

filelist = open(output_file,'a+')
filelist.write(input_file + '\n')
filelist.close()
```

Running METplus

This use case is run in the following ways:

1) Passing in UserScript_obsERA_obsOnly_PhaseDiagram.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_PhaseDiagram.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_obsERA_obsOnly_PhaseDiagram.py:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_PhaseDiagram.conf
```

The following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/s2s_mjo/UserScript_obsERA_obsOnly_PhaseDiagram. This may include the regridded data and daily averaged files. In addition, the phase diagram plots will be generated and the output location can be specified as PHASE_DIAGRAM_PLOT_OUTPUT_DIR. If it is not specified, plots will be sent to model_applications/s2s_mjo/UserScript_obsERA_obsOnly_PhaseDiagram/plots (relative to **OUTPUT_BASE**).

Keywords

Note:

- S2SAppUseCase
- S2SMJOAppUseCase
- METplotpyUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1997) to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s_mjo-UserScript_obsERA_obsOnly_PhaseDiagram.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.12.2 UserScript: Make OMI plot from calculated MJO indices

model_applications/ s2s_mjo/ UserScript_obsERA_obsOnly_OMI.py

Scientific Objective

To use Outgoing Longwave Radiation (OLR) to compute the OLR based MJO Index (OMI). Specifically, OMI is computed using OLR data between 20N and 20S. The OLR data are then projected onto Empirical Orthogonal Function (EOF) data that is computed for each day of the year, latitude, and longitude. The OLR is then filtered for 20 - 96 days, and regressed onto the daily EOFs. Finally, it's normalized and these normalized components are plotted on a phase diagram.

Datasets

- Forecast dataset: None
- Observation dataset: ERA Reanlaysis Outgoing Longwave Radiation.

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* numpy
* netCDF4
* datetime
* xarray
* matplotlib
* scipy
* pandas
```

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case runs the OMI driver which computes OMI and creates a phase diagram. Inputs to the OMI driver include netCDF files that are in MET's netCDF version. In addition, a txt file containing the listing of these input netCDF files is required, as well as text file listings of the EOF1 and EOF2 files. These text files can be generated using the USER_SCRIPT_INPUT_TEMPLATES in the [create_eof_filelist] and [script_omi] sections. Some optional pre-processing steps include using regrid_data_plane to either regrid your data or cut the domain to 20N - 20S.

METplus Workflow

The OMI driver script python code is run for each lead time on the forecast and observations data. This example loops by valid time for the model pre-processing, and valid time for the other steps. This version is set to only process the OMI calculation and creating a text file listing of the EOF files, omitting the creation of daily means for the model and the regridding pre-processing steps. However, the configurations for pre-processing are available for user reference.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_OMI.conf. The file UserScript_obsERA_obsOnly_OMI/OMI_driver.py runs the python program and UserScript_fcstGFS_obsERA_OMI.conf sets the variables for all steps of the OMI use case.

```
# OMI UserScript wrapper
[config]


# All steps, including pre-processing:
#PROCESS_LIST = RegridDataPlane(regrid_obs_olr), UserScript(create_eof_filelist),␣
↪UserScript(script_omi)
# Finding EOF files and OMI Analysis script for the observations
PROCESS_LIST = UserScript(create_eof_filelist), UserScript(script_omi)


LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 1979010100
VALID_END = 2012123000
VALID_INCREMENT = 86400

LEAD_SEQ = 0

# variables referenced in other sections
```

(continues on next page)

(continued from previous page)

```
# Run the obs for these cases
OBS_RUN = True
FCST_RUN = False

OBS_OLR_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_OMI/ERA
OBS_OLR_INPUT_TEMPLATE = OLR_{valid?fmt=%Y%m%d}.nc


###
# RegridDataPlane(regrid_obs_olr) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

REGRID_DATA_PLANE_VERIF_GRID = latlon 144 17 -20 0 2.5 2.5

REGRID_DATA_PLANE_METHOD = NEAREST

REGRID_DATA_PLANE_WIDTH = 1


###
# RegridDataPlane(regrid_obs_olr) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Configurations for regrid_data_plane:  Regrid OLR to -20 to 20 latitude
[regrid_obs_olr]

OBS_REGRID_DATA_PLANE_RUN = {OBS_RUN}

OBS_DATA_PLANE_ONCE_PER_FIELD = False

OBS_REGRID_DATA_PLANE_VAR1_NAME = olr
OBS_REGRID_DATA_PLANE_VAR1_LEVELS = "({valid?fmt=%Y%m%d_%H%M%S},*,*)"
OBS_REGRID_DATA_PLANE_VAR1_OPTIONS = file_type=NETCDF_NCCF; censor_thresh=eq-999.0; censor_
↪val=-9999.0;

OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = olr

OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_
↪obsOnly_OMI
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OBS_OLR_INPUT_DIR}

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = olr.1x.7920.nc
```

(continues on next page)

(continued from previous page)

```
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {OBS_OLR_INPUT_TEMPLATE}


###
# UserScript(create_eof_filelist) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Create the EOF filelists
[create_eof_filelist]

# Find the files for each time to create the time list
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE

# Valid Begin and End Times for the EOF files
VALID_BEG = 2012010100
VALID_END = 2012123100

# Find the EOF files for each time
# Filename templates for EOF1 and EOF2
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_
↪obsOnly_OMI/EOF/eof1/eof{valid?fmt=%j}.txt,{INPUT_BASE}/model_applications/s2s_mjo/
↪UserScript_obsERA_obsOnly_OMI/EOF/eof2/eof{valid?fmt=%j}.txt

# Name of the file containing the listing of input files
# The options are EOF1_INPUT and EOF2_INPUT
# *** Make sure the order is the same as the order of templates listed in USER_SCRIPT_INPUT_
↪TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS = EOF1_INPUT, EOF2_INPUT

# Placeholder command just to build the file list
# This just states that it's building the file list
USER_SCRIPT_COMMAND = echo Populated file list for EOF1 and EOF2 Input


# Configurations for the OMI analysis script
[user_env_vars]
# Whether to Run the model or obs
RUN_OBS = {OBS_RUN}
RUN_FCST = {FCST_RUN}

# Make OUTPUT_BASE Available to the script
SCRIPT_OUTPUT_BASE = {OUTPUT_BASE}

# Number of obs per day
```

(continues on next page)

(continued from previous page)

```
OBS_PER_DAY = 1

# Output Directory for the plots
# If not set, it this will default to {OUTPUT_BASE}/plots
OMI_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_OMI/plots

# Phase Plot start date, end date, output name, and format
PHASE_PLOT_TIME_BEG = 2012010100
PHASE_PLOT_TIME_END = 2012033000
PHASE_PLOT_TIME_FMT = {VALID_TIME_FMT}
OBS_PHASE_PLOT_OUTPUT_NAME = obs_OMI_comp_phase
OBS_PHASE_PLOT_OUTPUT_FORMAT = png


###
# UserScript(script_omi) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Configurations for UserScript: Run the RMM Analysis driver
[script_omi]
#  Run the script once per lead time
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

## Template of OLR filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {OBS_OLR_INPUT_DIR}/{OBS_OLR_INPUT_TEMPLATE}

## Name of the file containing the listing of OLR input files
## The options are OBS_OLR_INPUT and  FCST_OLR_INPUT
## *** Make sure the order is the same as the order of templates listed in USER_SCRIPT_INPUT_
↪TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS = OBS_OLR_INPUT

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_OMI/OMI_driver.py
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

Python Scripts

The OMI driver script orchestrates the calculation of the MJO indices and the generation of a phase diagram OMI plot: parm/use_cases/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_OMI/OMI_driver.py:

```
#!/usr/bin/env python3
"""
Driver Script to Compute RMM index from input U850, U200 and OLR data. Data is averaged from␣
↪20S-20N
"""

import numpy as np
import xarray as xr
import pandas as pd
import datetime
import glob
import os
import warnings

import metcalcpy.contributed.rmm_omi.compute_mjo_indices as cmi
import metplotpy.contributed.mjo_rmm_omi.plot_mjo_indices as pmi
import METreadnc.util.read_netcdf as read_netcdf


def read_omi_eofs(eof1_files, eof2_files):
    """
    Read the OMI EOFs from file and into a xarray DataArray.
    :param eofpath: filepath to the location of the eof files
    :return: EOF1 and EOF2 3D DataArrays
    """

    # observed EOFs from NOAA PSL are saved in individual text files for each doy
    # horizontal resolution of EOFs is 2.5 degree
    EOF1 = xr.DataArray(np.empty([366,17,144]),dims=['doy','lat','lon'],
    coords={'doy':np.arange(1,367,1), 'lat':np.arange(-20,22.5,2.5), 'lon':np.arange(0,360,2.
↪5)})
```

(continues on next page)

(continued from previous page)

```
    EOF2 = xr.DataArray(np.empty([366,17,144]),dims=['doy','lat','lon'],
    coords={'doy':np.arange(1,367,1), 'lat':np.arange(-20,22.5,2.5), 'lon':np.arange(0,360,2.
↪5)})
    nlat = len(EOF1['lat'])
    nlon = len(EOF1['lon'])

    for doy in range(len(eof1_files)):
        doystr = str(doy).zfill(3)
        tmp1 = pd.read_csv(eof1_files[doy], header=None, delim_whitespace=True, names=['eof1
↪'])
        tmp2 = pd.read_csv(eof2_files[doy], header=None, delim_whitespace=True, names=['eof2
↪'])
        eof1 = xr.DataArray(np.reshape(tmp1.eof1.values,(nlat, nlon)),dims=['lat','lon'])
        eof2 = xr.DataArray(np.reshape(tmp2.eof2.values,(nlat, nlon)),dims=['lat','lon'])
        EOF1[doy,:,:] = eof1.values
        EOF2[doy,:,:] = eof2.values

    return EOF1, EOF2


def run_omi_steps(inlabel, olr_filetxt, spd, EOF1, EOF2, oplot_dir):

    # Read the listing of EOF files
    with open(olr_filetxt) as ol:
        olr_input_files = ol.read().splitlines()
    if (olr_input_files[0] == 'file_list'):
        olr_input_files = olr_input_files[1:]

    # Read in the netCDF data from a list of files

    netcdf_reader = read_netcdf.ReadNetCDF()
    ds_orig = netcdf_reader.read_into_xarray(olr_input_files)

    # Add some needed attributes
    ds_list = []
    time = []
    for din in ds_orig:
        ctime =  datetime.datetime.strptime(din['olr'].valid_time,'%Y%m%d_%H%M%S')
        time.append(ctime.strftime('%Y-%m-%d'))
        din = din.assign_coords(time=ctime)
        din = din.expand_dims("time")
        ds_list.append(din)
    time = np.array(time,dtype='datetime64[D]')

    everything = xr.concat(ds_list,"time")
```

(continues on next page)

(continued from previous page)

```
    olr = everything['olr']
    print(olr.min(), olr.max())

    # project OLR onto EOFs
    PC1, PC2 = cmi.omi(olr, time, spd, EOF1, EOF2)

    # Get times for the PC phase diagram
    plase_plot_time_format = os.environ['PHASE_PLOT_TIME_FMT']
    phase_plot_start_time = datetime.datetime.strptime(os.environ['PHASE_PLOT_TIME_BEG'],
↪plase_plot_time_format)
    phase_plot_end_time = datetime.datetime.strptime(os.environ['PHASE_PLOT_TIME_END'],plase_
↪plot_time_format)
    PC1_plot = PC1.sel(time=slice(phase_plot_start_time,phase_plot_end_time))
    PC2_plot = PC2.sel(time=slice(phase_plot_start_time,phase_plot_end_time))

    # Get the output name and format for the PC plase diagram
    phase_plot_name = os.path.join(oplot_dir,os.environ.get(inlabel+'_PHASE_PLOT_OUTPUT_NAME
↪',inlabel+'_OMI_comp_phase'))
    print(phase_plot_name)
    phase_plot_format = os.environ.get(inlabel+'_PHASE_PLOT_OUTPUT_FORMAT','png')

    # plot the PC phase diagram
    pmi.phase_diagram('OMI',PC1,PC2,np.array(PC1_plot['time'].dt.strftime("%Y-%m-%d").
↪values),
        np.array(PC1_plot['time.month'].values),np.array(PC1_plot['time.day'].values),
        phase_plot_name,phase_plot_format)


def main():

    # Get Obs and Forecast OLR file listing
    obs_olr_filetxt = os.environ.get('METPLUS_FILELIST_OBS_OLR_INPUT','')
    fcst_olr_filetxt = os.environ.get('METPLUS_FILELIST_FCST_OLR_INPUT','')

    # Read in EOF filenames
    eof1_filetxt = os.environ['METPLUS_FILELIST_EOF1_INPUT']
    eof2_filetxt = os.environ['METPLUS_FILELIST_EOF2_INPUT']

    # Read the listing of EOF files
    with open(eof1_filetxt) as ef1:
        eof1_input_files = ef1.read().splitlines()
    if (eof1_input_files[0] == 'file_list'):
        eof1_input_files = eof1_input_files[1:]
    with open(eof2_filetxt) as ef2:
        eof2_input_files = ef2.read().splitlines()
```

(continues on next page)

(continued from previous page)

```
    if (eof2_input_files[0] == 'file_list'):
        eof2_input_files = eof2_input_files[1:]

    # Read in the EOFs
    EOF1, EOF2 = read_omi_eofs(eof1_input_files, eof2_input_files)

    # Get Number of Obs per day
    spd = os.environ.get('OBS_PER_DAY',1)

    # Check for an output plot directory in the configs.  Create one if it does not exist
    oplot_dir = os.environ.get('OMI_PLOT_OUTPUT_DIR','')
    if not oplot_dir:
        obase = os.environ['SCRIPT_OUTPUT_BASE']
        oplot_dir = os.path.join(obase,'plots')
    if not os.path.exists(oplot_dir):
        os.makedirs(oplot_dir)

    #  Determine if doing forecast or obs
    run_obs_omi = os.environ.get('RUN_OBS','False').lower()
    run_fcst_omi = os.environ.get('RUN_FCST', 'False').lower()

    # Run the steps to compute OMM
    # Observations
    if run_obs_omi == 'true':
        run_omi_steps('OBS', obs_olr_filetxt, spd, EOF1, EOF2, oplot_dir)

    # Forecast
    if run_fcst_omi == 'true':
        run_omi_steps('FCST', fcst_olr_filetxt, spd, EOF1, EOF2, oplot_dir)

    # nothing selected
    if (run_obs_omi == 'false') and (run_fcst_omi == 'false'):
        warnings.warn('Forecast and Obs runs not selected, nothing will be calculated')
        warnings.warn('Set RUN_FCST or RUN_OBS in the [user_en_vars] section to generate␣
↪output')


if __name__ == "__main__":
    main()
```

Running METplus

This use case is run in the following ways:

1) Passing in UserScript_obsERA_obsOnly_OMI.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_OMI.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_obsERA_obsOnly_OMI.py:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_OMI.conf
```

The following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/s2s_mjo/UserScript_obsERA_obsOnly_OMI. This may include the regridded data and daily averaged files. In addition, the phase diagram plots will be generated and the output location can be specified as OMI_PLOT_OUTPUT_DIR. If it is not specified, plots will be sent to model_applications/s2s_mjo/UserScript_obsERA_obsOnly_OMI/plots (relative to **OUTPUT_BASE**).

Keywords

Note:

- S2SAppUseCase
- S2SMJOAppUseCase
- RegridDataPlaneUseCase
- PCPCombineUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1997) to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s_mjo-UserScript_obsERA_obsOnly_OMI.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.12.3 UserScript: Make MaKE-MaKI plot from calculated MaKE and MaKI indices

model_applications/ s2s_mjo/ UserScript_obsCFSR_obsOnly_MJO_ENSO.py

Scientific Objective

To compute the MJO-Kelvin wave-ENSO (MaKE) and MJO-Kelvin wave-Influence (MaKI) indices* using the zonal and meridional components of winds tress (TAUX,TAUY), zonal and meridional components of surface ocean currents (UCUR,VCUR), and sea surface temperature (SST). Specifically, MaKE and MaKI indices are computed using TAUX, TAUY, UCUR, VCUR and SST data between 30S and 30N and 125E and 80W. Daily anomalies of wind stress components are filtered for 30-90 days using a Convolutional Neural Network (CNN)-based filter. The weights of the filter are computed offline. The bandpass filtered wind stress components are projected onto 4 Empirical Orthogonal Functions (EOFs) data. The obtained timeseries (PCs) are standardized and combined with the EOFs to obtain the MJO component of the surface wind stress (TAUX_MJO,TAUY_MJO). UCUR and VCUR daily anomalies are multiplied by the meridional structure of Kelvin wave (UCUR_K,VCUR_K). Windpower due to the MJO component of the wind stress and oceanic Kelvin waves (W_MJO,K) is then computed as TAUX_MJO*UCUR_K+TAUY_MJO*VCUR_K. The standardized windpower and SST are projected onto the first two multivariate EOFs of W_MJO,K and SST. The resulting daily time series (PCs) are normalized and used to compute monthly values of MaKE and MaKI. Monthly values of MaKE and MaKI are saved into a text (.csv) file and plotted as time series.

- Lybarger, N.D., C.-S. Shin, and C. Stan, 2020: MJO Wind energy and prediction of El Nino, Journal of Geophysical Research - Oceans, 125, e2020JC016732. doi:10.1029/2020JC016732

Datasets

- Forecast dataset: None
- Observation dataset: CFSR Reanalysis

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* numpy
* netCDF4
* datetime
* xarray
* matplotlib
* pandas
```

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case runs the MJO-ENSO driver, which first computes the MJO components of taux and tauy, then the MJO wind power, the MJO-ENSO indices, their plot. Inputs to the MJO-ENSO driver include netCDF files that are in MET's netCDF version. In addition, a text file containing the listing of these input netCDF files for taux, tauy, u, v, and SST is required. Some optional pre-processing steps include RegridDataPlane for regridding the data.

METplus Workflow

The MJO-ENSO driver script python code is run for each lead time on the forecast and observations data. This example loops by valid time for the model pre-processing, and valid time for the other steps. This version is set to only process the regridding, and MaKE and MaKI calculation, omitting the caluclation of the mean daily annucal cycle and daily anomalies pre-processing steps. However, the configurations for pre-processing are available for user reference.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO.conf. The file UserScript_obsCFSR_obsOnly_MJO_ENSO/mjo_enso_driver.py runs the python program and UserScript_obsCFSR_obsOnly_MJO_ENSO.conf sets the variables for all steps of the MJO-ENSO use case.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s_mjo/UserScript_
↪obsCFSR_obsOnly_MJO_ENSO.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

# All steps, including creating daily means and mean daily annual cycle
#PROCESS_LIST = RegridDataPlane(regrid_obs_taux), RegridDataPlane(regrid_obs_tauy),␣
↪RegridDataPlane(regrid_obs_sst), RegridDataPlane(regrid_obs_ucur),  RegridDataPlane(regrid_
↪obs_vcur), UserScript(script_mjo_enso)
# Computing regridding, and MJO ENSO Analysis script
#PROCESS_LIST = RegridDataPlane(regrid_obs_taux), RegridDataPlane(regrid_obs_tauy),␣
↪RegridDataPlane(regrid_obs_sst), RegridDataPlane(regrid_obs_ucur),  RegridDataPlane(regrid_
↪obs_vcur), UserScript(script_mjo_enso)

PROCESS_LIST = UserScript(script_mjo_enso)


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###
```

(continues on next page)

(continued from previous page)

```
LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG = 19900101
VALID_END = 20211231
VALID_INCREMENT = 86400

LEAD_SEQ = 0


# Run the obs for these cases
OBS_RUN = True
FCST_RUN = False


###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Mask to use for regridding
REGRID_DATA_PLANE_VERIF_GRID = latlon 156 61 -30 125 1 1

# Method to run regrid_data_plane, not setting this will default to NEAREST
REGRID_DATA_PLANE_METHOD = NEAREST

# Regridding width used in regrid_data_plane, not setting this will default to 1
REGRID_DATA_PLANE_WIDTH = 1


###
# RegridDataPlane(regrid_obs_taux) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Configurations for regrid_data_plane: Regrid OLR to -15 to 15 latitude
[regrid_obs_taux]
# Run regrid_data_plane on forecast data
OBS_REGRID_DATA_PLANE_RUN = {OBS_RUN}

# If true, process each field individually and write a file for each
# If false, run once per run time passing in all fields specified
REGRID_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_NAME = uflx
```

(continues on next page)

(continued from previous page)

```
# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = uflx

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR ={INPUT_BASE}/zonalWindStress/
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/
↪Regrid/zonalWindStress/

# format of filenames
# Input CFSR
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = cfsr_zonalWindStress_{valid?fmt=%Y%m%d}.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE =cfsr_zonalWindStress_{valid?fmt=%Y%m%d}.nc


###
# RegridDataPlane(regrid_obs_tauy) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Configurations for regrid_data_plane: Regrid meridional wind stress
[regrid_obs_tauy]
# Run regrid_data_plane on forecast data
OBS_REGRID_DATA_PLANE_RUN = {OBS_RUN}

# If true, process each field individually and write a file for each
# If false, run once per run time passing in all fields specified
REGRID_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_NAME = vflx

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = vflx

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR ={INPUT_BASE}/meridionalWindStress/
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/
↪Regrid/meridionalWindStress/

# format of filenames
# Input CFSR
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = cfsr_meridionalWindStress_{valid?fmt=%Y%m%d}.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = cfsr_meridionalWindStress_{valid?fmt=%Y%m%d}.nc
```

(continues on next page)

(continued from previous page)

```
###
# RegridDataPlane(regrid_obs_sst) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Configurations for regrid_data_plane: Regrid sst
[regrid_obs_sst]
# Run regrid_data_plane on forecast data
OBS_REGRID_DATA_PLANE_RUN = {OBS_RUN}

# If true, process each field individually and write a file for each
# If false, run once per run time passing in all fields specified
REGRID_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_NAME =sst

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = sst

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR ={INPUT_BASE}/sst/
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/
↪Regrid/sst/

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = cfsr_sst_{valid?fmt=%Y%m%d}.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = cfsr_sst_{valid?fmt=%Y%m%d}.nc


###
# RegridDataPlane(regrid_obs_ucur) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Configurations for regrid_data_plane: Regrid zonal ocean current
[regrid_obs_ucur]
# Run regrid_data_plane on forecast data
OBS_REGRID_DATA_PLANE_RUN = {OBS_RUN}

# If true, process each field individually and write a file for each
# If false, run once per run time passing in all fields specified
REGRID_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
```

(continues on next page)

(continued from previous page)

```
OBS_REGRID_DATA_PLANE_VAR1_NAME = u

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = u

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR ={INPUT_BASE}/zonalOceanCurrent/
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/
↪Regrid/zonalOceanCurrent/

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = cfsr_zonalOceanCurrent_{valid?fmt=%Y%m%d}.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = cfsr_zonalOceanCurrent_{valid?fmt=%Y%m%d}.nc



###
# RegridDataPlane(regrid_obs_vcur) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Configurations for regrid_data_plane: Regrid meridional ocean current
[regrid_obs_vcur]
# Run regrid_data_plane on forecast data
OBS_REGRID_DATA_PLANE_RUN = {OBS_RUN}

# If true, process each field individually and write a file for each
# If false, run once per run time passing in all fields specified
REGRID_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_NAME = v

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = v

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR ={INPUT_BASE}/meridionalOceanCurrent/
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/
↪Regrid/meridionalOceanCurrent/

# format of filenames
# Input CFSR
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = cfsr_meridionalOceanCurrent_{valid?fmt=%Y%m%d}.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = cfsr_meridionalOceanCurrent_{valid?fmt=%Y%m%d}.nc
```

(continues on next page)

(continued from previous page)

```
###
# UserScript(script_mjo_enso) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Configurations for UserScript: Run the MJO_ENSO Analysis driver
[script_mjo_enso]
# list of strings to loop over for each run time.
# Run the user script once per lead
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

# Template of filenames to input to the user-script
#USER_SCRIPT_INPUT_TEMPLATE = {OUTPUT_BASE}/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/
↪Regrid/zonalWindStress/cfsr_zonalWindStress_{valid?fmt=%Y%m%d}.nc,{OUTPUT_BASE}/s2s_mjo/
↪UserScript_obsCFSR_obsOnly_MJO_ENSO/Regrid/meridionalWindStress/cfsr_meridionalWindStress_
↪{valid?fmt=%Y%m%d}.nc,{OUTPUT_BASE}/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/Regrid/sst/
↪cfsr_sst_{valid?fmt=%Y%m%d}.nc,{OUTPUT_BASE}/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/
↪Regrid/zonalOceanCurrent/cfsr_zonalOceanCurrent_{valid?fmt=%Y%m%d}.nc,{OUTPUT_BASE}/s2s_
↪mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/Regrid/meridionalOceanCurrent/cfsr_
↪meridionalOceanCurrent_{valid?fmt=%Y%m%d}.nc

USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsCFSR_
↪obsOnly_MJO_ENSO/zonalWindStress/cfsr_zonalWindStress_{valid?fmt=%Y%m%d}.nc,{INPUT_BASE}/
↪model_applications/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/meridionalWindStress/cfsr_
↪meridionalWindStress_{valid?fmt=%Y%m%d}.nc,{INPUT_BASE}/model_applications/s2s_mjo/
↪UserScript_obsCFSR_obsOnly_MJO_ENSO/sst/cfsr_sst_{valid?fmt=%Y%m%d}.nc,{INPUT_BASE}/model_
↪applications/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/zonalOceanCurrent/cfsr_
↪zonalOceanCurrent_{valid?fmt=%Y%m%d}.nc,{INPUT_BASE}/model_applications/s2s_mjo/UserScript_
↪obsCFSR_obsOnly_MJO_ENSO/meridionalOceanCurrent/cfsr_meridionalOceanCurrent_{valid?fmt=%Y%m
↪%d}.nc

# Name of the file containing the listing of input files
# The options are OBS_TAUX_INPUT, OBS_TAUY_INPUT, OBS_SST_INPUT, OBS_UCUR_INPUT, OBS_VCUR_
↪INPUT, FCST_TAUX_INPUT, FCST_TAUY_INPUT, FCST_SST_INPUT, FCST_UCUR_INPUT,and FCST_VCUR_
↪INPUT
# *** Make sure the order is the same as the order of templates listed in USER_SCRIPT_INPUT_
↪TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS = OBS_TAUX_INPUT,OBS_TAUY_INPUT, OBS_SST_INPUT, OBS_UCUR_
↪INPUT, OBS_VCUR_INPUT

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsCFSR_obsOnly_MJO_ENSO/mjo_enso_driver.py
```

(continues on next page)

(continued from previous page)

```
# Configurations for the MJO-ENSO analysis script
[user_env_vars]
# Whether to Run the model or obs
RUN_OBS = {OBS_RUN}
RUN_FCST = {FCST_RUN}

# Make OUTPUT_BASE Available to the script
SCRIPT_OUTPUT_BASE = {OUTPUT_BASE}

# Number of obs per day
OBS_PER_DAY = 1

# Variable names for TAUX, TAUY, SST, UCUR, VCUR
OBS_TAUX_VAR_NAME = uflx
OBS_TAUY_VAR_NAME = vflx
OBS_SST_VAR_NAME = sst
OBS_UCUR_VAR_NAME = u
OBS_VCUR_VAR_NAME = v

# EOF Filename
TAUX_EOF_INPUT_FILE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_
↪ENSO/Data/cfs_uflx_eof.nc
TAUY_EOF_INPUT_FILE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_
↪ENSO/Data/cfs_vflx_eof.nc
WMJOK_SST_EOF_INPUT_FILE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsCFSR_
↪obsOnly_MJO_ENSO/Data/cfs_multivarEOF.nc

# Filters weights
TAUX_Filter1_TEXTFILE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsCFSR_obsOnly_
↪MJO_ENSO/Data/taux.filter1.txt
TAUX_Filter2_TEXTFILE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsCFSR_obsOnly_
↪MJO_ENSO/Data/taux.filter2.txt
TAUY_Filter1_TEXTFILE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsCFSR_obsOnly_
↪MJO_ENSO/Data/tauy.filter1.txt
TAUY_Filter2_TEXTFILE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsCFSR_obsOnly_
↪MJO_ENSO/Data/tauy.filter2.txt


# Output Directory for the plots
# If not set, it this will default to {OUTPUT_BASE}/plots
PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/plots

# MaKE, MaKI indices output file
MAKE_MAKI_OUTPUT_TEXT_FILE = {OUTPUT_BASE}/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/MAKE-
↪MAKI
```

(continues on next page)

(continued from previous page)

```
# Plot start date, end date, output name, and format
PLOT_TIME_BEG = 19900101
PLOT_TIME_END = 20211231
PLOT_TIME_FMT = {VALID_TIME_FMT}
OBS_PLOT_OUTPUT_NAME = MAKE_MAKI_timeseries
OBS_PLOT_OUTPUT_FORMAT = png
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

Python Scripts

The MJO-ENSO driver script orchestrates the calculation of the MaKE and MaKI indices and the generation of a text file and a plot for the indices: parm/use_cases/model_applications/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/mjo_enso_driver.py:

```
#!/usr/bin/env python3

import xarray as xr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import datetime
import warnings

import metcalcpy.contributed.mjo_enso.compute_mjo_enso as mj
import metplotpy.contributed.mjo_enso.plot_mjo_enso_indices as plt
import METreadnc.util.read_netcdf as read_netcdf


def read_eofs(taux_eofs_file, tauy_eofs_file, meofs_file):

    taux_eofs=xr.open_dataset(taux_eofs_file).eof
```

(continues on next page)

(continued from previous page)

```
    tauy_eofs=xr.open_dataset(tauy_eofs_file).eof
    meofs = xr.open_dataset(meofs_file).meofs

    return taux_eofs,tauy_eofs,meofs

def read_filters(filtx1fil,filtx2fil,filty1fil,filty2fil):
    filtx1=np.loadtxt(filtx1fil, delimiter=',')
    filtx2=np.loadtxt(filtx2fil, delimiter=',')
    filty1=np.loadtxt(filty1fil, delimiter=',')
    filty2=np.loadtxt(filty2fil, delimiter=',')

    return filtx1,filtx2,filty1,filty2

def run_mjo_enso_steps(inlabel,spd,filtx1,filtx2,filty1,filty2,taux_eofs,tauy_eofs,meofs,
↪oplot_dir):

    # Get TAUX, TAUY, SST, UCURRENT, VCURRENT file listings and variable names
    taux_filetxt = os.environ['METPLUS_FILELIST_'+inlabel+'_TAUX_INPUT']
    tauy_filetxt = os.environ['METPLUS_FILELIST_'+inlabel+'_TAUY_INPUT']
    sst_filetxt = os.environ['METPLUS_FILELIST_'+inlabel+'_SST_INPUT']
    ucur_filetxt = os.environ['METPLUS_FILELIST_'+inlabel+'_UCUR_INPUT']
    vcur_filetxt = os.environ['METPLUS_FILELIST_'+inlabel+'_VCUR_INPUT']

    taux_var = os.environ[inlabel+'_TAUX_VAR_NAME']
    tauy_var = os.environ[inlabel+'_TAUY_VAR_NAME']
    sst_var = os.environ[inlabel+'_SST_VAR_NAME']
    u_var = os.environ[inlabel+'_UCUR_VAR_NAME']
    v_var = os.environ[inlabel+'_VCUR_VAR_NAME']

    # Read the listing of TAUX, TAUY, SST, UCUR, VCUR files
    with open(taux_filetxt) as tx:
        taux_input_files = tx.read().splitlines()
    if (taux_input_files[0] == 'file_list'):
        taux_input_files = taux_input_files[1:]
    with open(tauy_filetxt) as ty:
        tauy_input_files = ty.read().splitlines()
    if (tauy_input_files[0] == 'file_list'):
        tauy_input_files = tauy_input_files[1:]
    with open(sst_filetxt) as ts:
        sst_input_files = ts.read().splitlines()
    if (sst_input_files[0] == 'file_list'):
        sst_input_files = sst_input_files[1:]
    with open(ucur_filetxt) as uc:
        ucur_input_files = uc.read().splitlines()
    if (ucur_input_files[0] == 'file_list'):
```

(continues on next page)

(continued from previous page)

```
    ucur_input_files = ucur_input_files[1:]
with open(vcur_filetxt) as vc:
    vcur_input_files = vc.read().splitlines()
if (vcur_input_files[0] == 'file_list'):
    vcur_input_files = vcur_input_files[1:]

# Check the input data to make sure it's not all missing
taux_allmissing = all(elem == 'missing' for elem in taux_input_files)
if taux_allmissing:
    raise IOError ('No input TAUX files were found, check file paths')
tauy_allmissing = all(elem == 'missing' for elem in tauy_input_files)
if tauy_allmissing:
    raise IOError('No input TUAY files were found, check file paths')
sst_allmissing = all(elem == 'missing' for elem in sst_input_files)
if sst_allmissing:
    raise IOError('No input SST files were found, check file paths')
ucur_allmissing = all(elem == 'missing' for elem in ucur_input_files)
if ucur_allmissing:
    raise IOError('No input UCUR files were found, check file paths')
vcur_allmissing = all(elem == 'missing' for elem in vcur_input_files)
if vcur_allmissing:
    raise IOError('No input VCUR files were found, check file paths')

netcdf_reader_taux=read_netcdf.ReadNetCDF()
ds_taux=netcdf_reader_taux.read_into_xarray(taux_input_files)

netcdf_reader_tauy=read_netcdf.ReadNetCDF()
ds_tauy=netcdf_reader_tauy.read_into_xarray(tauy_input_files)

netcdf_reader_sst=read_netcdf.ReadNetCDF()
ds_sst=netcdf_reader_sst.read_into_xarray(sst_input_files)

netcdf_reader_ucur=read_netcdf.ReadNetCDF()
ds_ucur=netcdf_reader_ucur.read_into_xarray(ucur_input_files)

netcdf_reader_vcur=read_netcdf.ReadNetCDF()
ds_vcur=netcdf_reader_vcur.read_into_xarray(vcur_input_files)

time = []
for din in range(len(ds_taux)):
    ctaux = ds_taux[din]
    #ctime =  datetime.datetime.strptime(ctaux[taux_var].valid_time,'%Y%m%d_%H%M%S')
    ctime =  datetime.datetime.strptime(str(ctaux['time'][0].values)[0:10],'%Y-%m-%d')
    time.append(ctime.strftime('%Y-%m-%d'))
    #ctaux = ctaux.assign_coords(time=ctime)
```

(continues on next page)

(continued from previous page)

```
        #ds_taux[din] = ctaux.expand_dims("time")

        ctauy = ds_tauy[din]
        #ctauy = ctauy.assign_coords(time=ctime)
        #ds_tauy[din] = ctauy.expand_dims("time")

        csst = ds_sst[din]
        #csst = csst.assign_coords(time=ctime)
        #ds_sst[din] = csst.expand_dims("time")

        cucur = ds_ucur[din]
        #cucur = cucur.assign_coords(time=ctime)
        #ds_ucur[din] = cucur.expand_dims("time")

        cvcur = ds_vcur[din]
        #cvcur = cvcur.assign_coords(time=ctime)
        #ds_vcur[din] = cvcur.expand_dims("time")

    time = np.array(time,dtype='datetime64[D]')

    everything_taux = xr.concat(ds_taux,"time")
    uflxa = everything_taux[taux_var]

    everything_tauy = xr.concat(ds_tauy,"time")
    vflxa = everything_tauy[tauy_var]

    everything_sst = xr.concat(ds_sst,"time")
    sst = everything_sst[sst_var]

    everything_ucur = xr.concat(ds_ucur,"time")
    u = everything_ucur[u_var]

    everything_vcur = xr.concat(ds_vcur,"time")
    v = everything_vcur[v_var]
    print(v.shape)

    # get taux_mjo and tauy_mjo

    uflx_mjo=mj.calc_tau_MJO(uflxa,taux_eofs,filtx1,filtx2)
    vflx_mjo=mj.calc_tau_MJO(vflxa,tauy_eofs,filty1,filty2)

    wpower=mj.calc_wpower_MJO(u,v,uflx_mjo,vflx_mjo)

    #sst = ds.sst.sel(lat=slice(-5,5)).mean(dim='lat',skipna=True)
    sst = sst.sel(lat=slice(-5,5)).mean(dim='lat',skipna=True)
```

(continues on next page)

(continued from previous page)

```
    wmjoks = wpower.sel(lat=slice(-5,5)).mean(dim='lat',skipna=True)

    make,maki=mj.make_maki(sst,wmjoks,meofs)

    #Get the index output file
    index_file = os.environ['MAKE_MAKI_OUTPUT_TEXT_FILE']
    import csv
    date_format = '%Y-%m-%d'
    strDate=datetime.datetime.strptime(str(sst['time'][0].values)[0:10],date_format)
    endDate=datetime.datetime.strptime(str(sst['time'][-1].values)[0:10],date_format)
    time_mon = pd.date_range(strDate, endDate, freq='MS')#.to_pydatetime().tolist()
    with open(index_file+'.csv', 'w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["Date", "MaKE", "MaKI"])
        for i in range(len(make)):
            writer.writerow([time_mon[i], make[i].data, maki[i].data])

    #Get times for plotting MaKE and MaKI indices
    plot_time_format = os.environ['PLOT_TIME_FMT']
    plot_start_time = datetime.datetime.strptime(os.environ['PLOT_TIME_BEG'],plot_time_
↪format)
    plot_end_time = datetime.datetime.strptime(os.environ['PLOT_TIME_END'],plot_time_format)

    make_plot = make.sel(time=slice(plot_start_time,plot_end_time))
    maki_plot = maki.sel(time=slice(plot_start_time,plot_end_time))

    # Get the output name and format for the MaKE and MaKi plot
    plot_name = os.path.join(oplot_dir,os.environ.get(inlabel+'_PLOT_OUTPUT_NAME',inlabel+'_
↪MAKE_MAKI_timeseries'))
    plot_format = os.environ.get(inlabel+'_PLOT_OUTPUT_FORMAT','png')

    #plot the MaKE-MaKI indices
    plt.plot_make_maki(make_plot,maki_plot,np.array(make_plot['time'].values),plot_name,plot_
↪format)


def main():

    # Get the EOF files
    taux_eofs_file = os.environ['TAUX_EOF_INPUT_FILE']
    tauy_eofs_file = os.environ['TAUY_EOF_INPUT_FILE']
    meofs_file = os.environ['WMJOK_SST_EOF_INPUT_FILE']

    # Read in the EOFS
```

(continues on next page)

(continued from previous page)

```
    print('Reading the EOFs')
    taux_eofs,tauy_eofs,meofs = read_eofs(taux_eofs_file, tauy_eofs_file, meofs_file)
    print('Done with reading EOFs')

    #Get the filter weights files
    filtx1fil = os.environ['TAUX_Filter1_TEXTFILE']
    filtx2fil = os.environ['TAUX_Filter2_TEXTFILE']
    filty1fil = os.environ['TAUY_Filter1_TEXTFILE']
    filty2fil = os.environ['TAUY_Filter2_TEXTFILE']

    # Read in the weights of the filters
    filtx1,filtx2,filty1,filty2 = read_filters(filtx1fil,filtx2fil,filtx2fil,filty2fil)

    # Get Number of Obs per day
    spd = os.environ.get('OBS_PER_DAY',1)

   # Check for an output plot directory
    oplot_dir = os.environ.get('PLOT_OUTPUT_DIR','')
    if not oplot_dir:
        obase = os.environ['SCRIPT_OUTPUT_BASE']
        oplot_dir = os.path.join(obase,'plots')
    if not os.path.exists(oplot_dir):
        os.makedirs(oplot_dir)

   # Determine if doing forecast or obs
    run_obs_mjo_enso = os.environ.get('RUN_OBS', 'False').lower()
    run_fcst_mjo_enso = os.environ.get('RUN_FCST', 'False').lower()

    if (run_obs_mjo_enso == 'true'):
        run_mjo_enso_steps('OBS', spd, filtx1, filtx2, filty1, filty2, taux_eofs, tauy_eofs,␣
↪meofs,oplot_dir)

    if (run_fcst_mjo_enso == 'true'):
        run_mjo_enso_steps('FCST', spd, filtx1, filtx2, filty1, filty2, taux_eofs, tauy_eofs,
↪ meofs,oplot_dir)

    # nothing selected
    if (run_obs_mjo_enso == 'false') and (run_fcst_mjo_enso == 'false'):
        warnings.warn('Forecast and Obs runs not selected, nothing will be calculated')
        warnings.warn('Set RUN_FCST or RUN_OBS in the [user_en_vars] section to generate␣
↪output')

if __name__ == "__main__":
    main()
```

Running METplus

This use case is run in the following ways:

1) Passing in UserScript_obsCFSR_obsOnly_MJO_ENSO.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsCFSR_obsOnly_MJO_ENSO.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_obsCFSR_obsOnly_MJO_ENSO.py:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsCFSR_obsOnly_MJO_ENSO.conf
```

The following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO. This may include the regridded data. In addition, a text (.csv) file will be generated and a time serie plot. The name of the text file can be specified as MAKE_MAKI_OUTPUT_TEXT_FILE. The output location can be specified as PLOT_OUTPUT_DIR. If it is not specified, plot will be sent to model_applications/s2s_mjo/UserScript_obsCFSR_obsOnly_MJO_ENSO/plots (relative to **OUTPUT_BASE**). The name of the plot file can be specified as OBS_PLOT_OUTPUT_NAME.

Keywords

Note:

- S2SAppUseCase
- S2SMJOAppUseCase
- NetCDFFileUseCase
- RegridDataPlaneUseCase
- PCPCombineUseCase
- METcalcpyUseCase
- METplotpyUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1997) to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s_mjo-UserScript_obsCFSr_obsOnly_MJO_ENSO.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.12.4 UserScript: Make OMI plot from calculated MJO indices

model_applications/ s2s_mjo/ UserScript_fcstGFS_obsERA_OMI.py

Scientific Objective

To use Outgoing Longwave Radiation (OLR) to compute the OLR based MJO Index (OMI). Specifically, OMI is computed using OLR data between 20N and 20S. The OLR data are then projected onto Empirical Orthogonal Function (EOF) data that is computed for each day of the year, latitude, and longitude. The OLR is then filtered for 20 - 96 days, and regressed onto the daily EOFs. Finally, it's normalized and these normalized components are plotted on a phase diagram. Separate phase diagrams are created for the model and observations.

Datasets

- Forecast dataset: GFS Model Outgoing Longwave Radiation
- Observation dataset: ERA Reanlaysis Outgoing Longwave Radiation.

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* numpy
* netCDF4
* datetime
* xarray
* matplotlib
* scipy
* pandas
```

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case runs the OMI driver which computes OMI and creates a phase diagram. Inputs to the OMI driver include netCDF files that are in MET's netCDF version. In addition, a txt file containing the listing of these input netCDF files is required, as well as text file listings of the EOF1 and EOF2 files. These text files can be generated using the USER_SCRIPT_INPUT_TEMPLATES in the [create_eof_filelist] and [script_omi] sections. Some optional pre-processing steps include using regrid_data_plane to either regrid your data or cut the domain to 20N - 20S.

METplus Workflow

The OMI driver script python code is run for each lead time on the forecast and observations data. This example loops by valid time for the model pre-processing, and valid time for the other steps. This version is set to only process the OMI calculation and creating a text file listing of the EOF files, omitting the creation of daily means for the model and the regridding pre-processing steps. However, the configurations for pre-processing are available for user reference.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/s2s_mjo/UserScript_fcstGFS_obsERA_OMI.conf. The file UserScript_fcstGFS_obsERA_OMI/OMI_driver.py runs the python program and UserScript_fcstGFS_obsERA_OMI.conf sets the variables for all steps of the OMI use case.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s_mjo/UserScript_
↪fcstGFS_obsERA_OMI.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

# All steps, including pre-processing:
#PROCESS_LIST = PcpCombine(daily_mean_fcst), RegridDataPlane(regrid_obs_olr),␣
↪RegridDataPlane(regrid_fcst_olr), UserScript(create_eof_filelist), UserScript(script_omi)
# Finding EOF files and OMI Analysis script for the observations

PROCESS_LIST = UserScript(create_eof_filelist), UserScript(script_omi)


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2017010100
VALID_END = 2018123100
VALID_INCREMENT = 86400
```

(continues on next page)

(continued from previous page)

```
LEAD_SEQ = 0

# variables referenced in other sections

# Run the obs for these cases
OBS_RUN = True
FCST_RUN = True

# Input and Output Directories for the OBS OLR Files and output text file containing the␣
↪file list
OBS_OLR_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_fcstGFS_obsERA_OMI/
↪ERA/Regrid
OBS_OLR_INPUT_TEMPLATE = OLR_{valid?fmt=%Y%m%d}.nc

# Input and Output Directories for the OBS OLR Files and output text file containing the␣
↪file list
FCST_OLR_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_fcstGFS_obsERA_OMI/
↪GFS/Regrid
FCST_OLR_INPUT_TEMPLATE = OLR_{valid?fmt=%Y%m%d}.nc


###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

REGRID_DATA_PLANE_VERIF_GRID = latlon 144 17 -20 0 2.5 2.5

REGRID_DATA_PLANE_METHOD = NEAREST

REGRID_DATA_PLANE_WIDTH = 1


###
# PCPCombine(daily_mean_fcst) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Configurations for pcp_combine:  Create daily means for the GFS
[daily_mean_fcst]

FCST_PCP_COMBINE_RUN = {FCST_RUN}

FCST_PCP_COMBINE_METHOD = USER_DEFINED
```

(continues on next page)

(continued from previous page)

```
FCST_PCP_COMBINE_COMMAND = -derive mean {FCST_PCP_COMBINE_INPUT_DIR}/{valid?fmt=%Y}/{valid?
↪fmt=%Y%m%d}/gfs.0p25.{valid?fmt=%Y%m%d%H}.f{lead?fmt=%HHH?shift=86400}.grib2 {FCST_PCP_
↪COMBINE_INPUT_DIR}/{valid?fmt=%Y}/{valid?fmt=%Y%m%d}/gfs.0p25.{valid?fmt=%Y%m%d%H}.f{lead?
↪fmt=%HHH?shift=75600}.grib2 {FCST_PCP_COMBINE_INPUT_DIR}/{valid?fmt=%Y}/{valid?fmt=%Y%m%d}/
↪gfs.0p25.{valid?fmt=%Y%m%d%H}.f{lead?fmt=%HHH?shift=64800}.grib2 {FCST_PCP_COMBINE_INPUT_
↪DIR}/{init?fmt=%Y}/{init?fmt=%Y%m%d}/gfs.0p25.{init?fmt=%Y%m%d%H}.f{lead?fmt=%HHH?
↪shift=54000}.grib2 {FCST_PCP_COMBINE_INPUT_DIR}/{init?fmt=%Y}/{init?fmt=%Y%m%d}/gfs.0p25.
↪{init?fmt=%Y%m%d%H}.f{lead?fmt=%HHH?shift=43200}.grib2 {FCST_PCP_COMBINE_INPUT_DIR}/{init?
↪fmt=%Y}/{init?fmt=%Y%m%d}/gfs.0p25.{init?fmt=%Y%m%d%H}.f{lead?fmt=%HHH?shift=32400}.grib2
↪{FCST_PCP_COMBINE_INPUT_DIR}/{init?fmt=%Y}/{init?fmt=%Y%m%d}/gfs.0p25.{init?fmt=%Y%m%d%H}.f
↪{lead?fmt=%HHH?shift=21600}.grib2 {FCST_PCP_COMBINE_INPUT_DIR}/{init?fmt=%Y}/{init?fmt=%Y%m
↪%d}/gfs.0p25.{init?fmt=%Y%m%d%H}.f{lead?fmt=%HHH?shift=10800}.grib2 -field 'name="ULWRF";␣
↪level="L0"; set_attr_valid = "{valid?fmt=%Y%m%d_%H%M%S}"; GRIB2_ipdtmpl_index = 9; GRIB2_
↪ipdtmpl_val = 8;'

FCST_PCP_COMBINE_INPUT_DIR = /gpfs/fs1/collections/rda/data/ds084.1
FCST_PCP_COMBINE_INPUT_TEMPLATE = {valid?fmt=%Y%m}/gfs.0p25.{init?fmt=%y%m%d%H}.f{lead?fmt=
↪%HHH}.grib2

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_fcstGFS_obsERA_OMI/GFS/daily_
↪mean
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = GFS_mean_{valid?fmt=%Y%m%d}.nc


###
# RegridDataPlane(regrid_obs_olr) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Configurations for regrid_data_plane:  Regrid ERA OLR to -20 to 20 latitude
[regrid_obs_olr]

LEAD_SEQ = 0

OBS_REGRID_DATA_PLANE_RUN = {OBS_RUN}

OBS_DATA_PLANE_ONCE_PER_FIELD = False

OBS_REGRID_DATA_PLANE_VAR1_NAME = olr
OBS_REGRID_DATA_PLANE_VAR1_LEVELS = "({valid?fmt=%Y%m%d_%H%M%S},*,*)"
OBS_REGRID_DATA_PLANE_VAR1_OPTIONS = file_type=NETCDF_NCCF; censor_thresh=eq-999.0; censor_
↪val=-9999.0;

OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = olr
```

(continues on next page)

(continued from previous page)

```
OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_fcstGFS_
↪obsERA_OMI/ERA/daily_mean
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OBS_OLR_INPUT_DIR}

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = olr.1x.7920.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {OBS_OLR_INPUT_TEMPLATE}


###
# RegridDataPlane(regrid_fcst_olr) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Configurations for regrid_data_plane:  Regrid GFS OLR to -20 to 20 latitude
[regrid_fcst_olr]

FCST_REGRID_DATA_PLANE_RUN = {FCST_RUN}

REGRID_DATA_PLANE_ONCE_PER_FIELD = False

FCST_REGRID_DATA_PLANE_VAR1_NAME = ULWRF_L0_mean
FCST_REGRID_DATA_PLANE_VAR1_LEVELS = "(*,*)"
FCST_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = olr

FCST_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_
↪fcstGFS_obsERA_OMI/GFS/daily_mean
FCST_REGRID_DATA_PLANE_OUTPUT_DIR = {FCST_OLR_INPUT_DIR}

FCST_REGRID_DATA_PLANE_INPUT_TEMPLATE = GFS_mean_{valid?fmt=%Y%m%d}.nc
FCST_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {FCST_OLR_INPUT_TEMPLATE}


###
# UserScript(create_eof_filelist) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Create the EOF filelists
[create_eof_filelist]

# Find the files for each time to create the time list
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE

# Valid Begin and End Times for the EOF files
```

(continues on next page)

(continued from previous page)

```
VALID_BEG = 2012010100
VALID_END = 2012123100

# Find the EOF files for each time
# Filename templates for EOF1 and EOF2
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_fcstGFS_
↪obsERA_OMI/EOF/eof1/eof{valid?fmt=%j}.txt,{INPUT_BASE}/model_applications/s2s_mjo/
↪UserScript_fcstGFS_obsERA_OMI/EOF/eof2/eof{valid?fmt=%j}.txt

# Name of the file containing the listing of input files
# The options are EOF1_INPUT and EOF2_INPUT
# *** Make sure the order is the same as the order of templates listed in USER_SCRIPT_INPUT_
↪TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS = EOF1_INPUT, EOF2_INPUT

# Placeholder command just to build the file list
# This just states that it's building the file list
USER_SCRIPT_COMMAND = echo Populated file list for EOF1 and EOF2 Input


# Configurations for the OMI analysis script
[user_env_vars]
# Whether to Run the model or obs
RUN_OBS = {OBS_RUN}
RUN_FCST = {FCST_RUN}

# Make OUTPUT_BASE Available to the script
SCRIPT_OUTPUT_BASE = {OUTPUT_BASE}

# Number of obs per day
OBS_PER_DAY = 1

# Output Directory for the plots
# If not set, it this will default to {OUTPUT_BASE}/plots
OMI_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_fcstGFS_obsERA_OMI/plots

# Phase Plot start date, end date, output name, and format
PHASE_PLOT_TIME_BEG = 2017010100
PHASE_PLOT_TIME_END = 2017033100
PHASE_PLOT_TIME_FMT = {VALID_TIME_FMT}
OBS_PHASE_PLOT_OUTPUT_NAME = obs_OMI_comp_phase
OBS_PHASE_PLOT_OUTPUT_FORMAT = png
FCST_PHASE_PLOT_OUTPUT_NAME = fcst_OMI_comp_phase
FCST_PHASE_PLOT_OUTPUT_FORMAT = png
```

(continues on next page)

(continued from previous page)

```
###
# UserScript(script_omi) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Configurations for UserScript: Run the RMM Analysis driver
[script_omi]
#  Run the script once per lead time
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

## Template of OLR filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {OBS_OLR_INPUT_DIR}/{OBS_OLR_INPUT_TEMPLATE},{FCST_OLR_INPUT_
↪DIR}/{FCST_OLR_INPUT_TEMPLATE}

## Name of the file containing the listing of OLR input files
## The options are OBS_OLR_INPUT and  FCST_OLR_INPUT
## *** Make sure the order is the same as the order of templates listed in USER_SCRIPT_INPUT_
↪TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS = OBS_OLR_INPUT,FCST_OLR_INPUT

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪fcstGFS_obsERA_OMI/OMI_driver.py
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

Python Scripts

The OMI driver script orchestrates the calculation of the MJO indices and the generation of a phase diagram OMI plot: parm/use_cases/model_applications/s2s_mjo/UserScript_fcstGFS_obsERA_OMI/OMI_driver.py:

```
#!/usr/bin/env python3
"""
Driver Script to Compute RMM index from input U850, U200 and OLR data. Data is averaged from␣
```

(continues on next page)

(continued from previous page)

```
↪20S-20N
"""

import numpy as np
import xarray as xr
import pandas as pd
import datetime
import glob
import os
import warnings

import metcalcpy.contributed.rmm_omi.compute_mjo_indices as cmi
import metplotpy.contributed.mjo_rmm_omi.plot_mjo_indices as pmi
import METreadnc.util.read_netcdf as read_netcdf


def read_omi_eofs(eof1_files, eof2_files):
    """
    Read the OMI EOFs from file and into a xarray DataArray.
    :param eofpath: filepath to the location of the eof files
    :return: EOF1 and EOF2 3D DataArrays
    """

    # observed EOFs from NOAA PSL are saved in individual text files for each doy
    # horizontal resolution of EOFs is 2.5 degree
    EOF1 = xr.DataArray(np.empty([366,17,144]),dims=['doy','lat','lon'],
    coords={'doy':np.arange(1,367,1), 'lat':np.arange(-20,22.5,2.5), 'lon':np.arange(0,360,2.
↪5)})
    EOF2 = xr.DataArray(np.empty([366,17,144]),dims=['doy','lat','lon'],
    coords={'doy':np.arange(1,367,1), 'lat':np.arange(-20,22.5,2.5), 'lon':np.arange(0,360,2.
↪5)})
    nlat = len(EOF1['lat'])
    nlon = len(EOF1['lon'])

    for doy in range(len(eof1_files)):
        doystr = str(doy).zfill(3)
        tmp1 = pd.read_csv(eof1_files[doy], header=None, delim_whitespace=True, names=['eof1
↪'])
        tmp2 = pd.read_csv(eof2_files[doy], header=None, delim_whitespace=True, names=['eof2
↪'])
        eof1 = xr.DataArray(np.reshape(tmp1.eof1.values,(nlat, nlon)),dims=['lat','lon'])
        eof2 = xr.DataArray(np.reshape(tmp2.eof2.values,(nlat, nlon)),dims=['lat','lon'])
        EOF1[doy,:,:] = eof1.values
        EOF2[doy,:,:] = eof2.values
```

(continues on next page)

(continued from previous page)

```
    return EOF1, EOF2


def run_omi_steps(inlabel, olr_filetxt, spd, EOF1, EOF2, oplot_dir):

    # Read the listing of EOF files
    with open(olr_filetxt) as ol:
        olr_input_files = ol.read().splitlines()
    if (olr_input_files[0] == 'file_list'):
        olr_input_files = olr_input_files[1:]

    # Read in the netCDF data from a list of files

    netcdf_reader = read_netcdf.ReadNetCDF()
    ds_orig = netcdf_reader.read_into_xarray(olr_input_files)

    # Add some needed attributes
    ds_list = []
    time = []
    for din in ds_orig:
        ctime =  datetime.datetime.strptime(din['olr'].valid_time,'%Y%m%d_%H%M%S')
        time.append(ctime.strftime('%Y-%m-%d'))
        din = din.assign_coords(time=ctime)
        din = din.expand_dims("time")
        ds_list.append(din)
    time = np.array(time,dtype='datetime64[D]')

    everything = xr.concat(ds_list,"time")
    olr = everything['olr']
    print(olr.min(), olr.max())

    # project OLR onto EOFs
    PC1, PC2 = cmi.omi(olr, time, spd, EOF1, EOF2)

    # Get times for the PC phase diagram
    plase_plot_time_format = os.environ['PHASE_PLOT_TIME_FMT']
    phase_plot_start_time = datetime.datetime.strptime(os.environ['PHASE_PLOT_TIME_BEG'],
↪plase_plot_time_format)
    phase_plot_end_time = datetime.datetime.strptime(os.environ['PHASE_PLOT_TIME_END'],plase_
↪plot_time_format)
    PC1_plot = PC1.sel(time=slice(phase_plot_start_time,phase_plot_end_time))
    PC2_plot = PC2.sel(time=slice(phase_plot_start_time,phase_plot_end_time))

    # Get the output name and format for the PC plase diagram
    phase_plot_name = os.path.join(oplot_dir,os.environ.get(inlabel+'_PHASE_PLOT_OUTPUT_NAME
```

(continues on next page)

(continued from previous page)

```
↪',inlabel+'_OMI_comp_phase'))
    print(phase_plot_name)
    phase_plot_format = os.environ.get(inlabel+'_PHASE_PLOT_OUTPUT_FORMAT','png')

    # plot the PC phase diagram
    pmi.phase_diagram('OMI',PC1,PC2,np.array(PC1_plot['time'].dt.strftime("%Y-%m-%d").
↪values),
        np.array(PC1_plot['time.month'].values),np.array(PC1_plot['time.day'].values),
        phase_plot_name,phase_plot_format)


def main():

    # Get Obs and Forecast OLR file listing
    obs_olr_filetxt = os.environ.get('METPLUS_FILELIST_OBS_OLR_INPUT','')
    fcst_olr_filetxt = os.environ.get('METPLUS_FILELIST_FCST_OLR_INPUT','')

    # Read in EOF filenames
    eof1_filetxt = os.environ['METPLUS_FILELIST_EOF1_INPUT']
    eof2_filetxt = os.environ['METPLUS_FILELIST_EOF2_INPUT']

    # Read the listing of EOF files
    with open(eof1_filetxt) as ef1:
        eof1_input_files = ef1.read().splitlines()
    if (eof1_input_files[0] == 'file_list'):
        eof1_input_files = eof1_input_files[1:]
    with open(eof2_filetxt) as ef2:
        eof2_input_files = ef2.read().splitlines()
    if (eof2_input_files[0] == 'file_list'):
        eof2_input_files = eof2_input_files[1:]

    # Read in the EOFs
    EOF1, EOF2 = read_omi_eofs(eof1_input_files, eof2_input_files)

    # Get Number of Obs per day
    spd = os.environ.get('OBS_PER_DAY',1)

    # Check for an output plot directory in the configs.  Create one if it does not exist
    oplot_dir = os.environ.get('OMI_PLOT_OUTPUT_DIR','')
    if not oplot_dir:
        obase = os.environ['SCRIPT_OUTPUT_BASE']
        oplot_dir = os.path.join(obase,'plots')
    if not os.path.exists(oplot_dir):
        os.makedirs(oplot_dir)
```

(continues on next page)

(continued from previous page)

```
    #  Determine if doing forecast or obs
    run_obs_omi = os.environ.get('RUN_OBS','False').lower()
    run_fcst_omi = os.environ.get('RUN_FCST', 'False').lower()

    # Run the steps to compute OMM
    # Observations
    if run_obs_omi == 'true':
        run_omi_steps('OBS', obs_olr_filetxt, spd, EOF1, EOF2, oplot_dir)

    # Forecast
    if run_fcst_omi == 'true':
        run_omi_steps('FCST', fcst_olr_filetxt, spd, EOF1, EOF2, oplot_dir)

    # nothing selected
    if (run_obs_omi == 'false') and (run_fcst_omi == 'false'):
        warnings.warn('Forecast and Obs runs not selected, nothing will be calculated')
        warnings.warn('Set RUN_FCST or RUN_OBS in the [user_en_vars] section to generate␣
↪output')


if __name__ == "__main__":
    main()
```

Running METplus

This use case is run in the following ways:

1) Passing in UserScript_fcstGFS_obsERA_OMI.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪fcstGFS_obsERA_OMI.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_fcstGFS_obsERA_OMI.py:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪fcstGFS_obsERA_OMI.conf
```

The following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/s2s_mjo/UserScript_fcstGFS_obsERA_OMI. This may include the regridded data and daily averaged files. In addition, the phase diagram plots will be generated and the output location can be specified as OMI_PLOT_OUTPUT_DIR. If it is not specified, plots will be sent to model_applications/s2s_mjo/UserScript_fcstGFS_obsERA_OMI/plots (relative to **OUTPUT_BASE**).

Keywords

Note:

- S2SAppUseCase
- S2SMJOAppUseCase
- RegridDataPlaneUseCase
- PCPCombineUseCase
- METcalcpyUseCase
- METplotpyUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1997) to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s_mjo-UserScript_fcstGFS_obsERA_OMI.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.12.5 UserScript: Make RMM plots from calculated MJO indices

model_applications/ s2s_mjo/ UserScript_obsERA_obsOnly_RMM.py

Scientific Objective

To compute the Real-time Multivariate MJO Index (RMM) using Outgoing Longwave Radiation (OLR), 850 hPa wind (U850), and 200 hPa wind (U200). Specifically, RMM is computed using OLR, U850, and U200 data between 15N and 15S. Anomalies of OLR, U850, and U200 are created using a harmonic analysis, 120 day day mean removed, and the data are normalized by normalization factors (generally the square root of the average variance) The anomalies are projected onto Empirical Orthogonal Function (EOF) data. The OLR is then filtered for 20 - 96 days, and regressed onto the daily EOFs. Finally, it's normalized and these normalized components are plotted on a phase diagram and timeseries plot.

Datasets

- Forecast dataset: None
- Observation dataset: ERA Reanlaysis Outgoing Longwave Radiation, 850 hPa wind and 200 hPa wind

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* numpy
* netCDF4
* datetime
* xarray
* matplotlib
* scipy
* pandas
```

If the version of Python used to compile MET did not have these libraries at the time of compilation, you will need to add these packages or create a new Python environment with these packages.

If this is the case, you will need to set the MET_PYTHON_EXE environment variable to the path of the version of Python you want to use. If you want this version of Python to only apply to this use case, set it in the [user_env_vars] section of a METplus configuration file.:

[user_env_vars] MET_PYTHON_EXE = /path/to/python/with/required/packages/bin/python

METplus Components

This use case runs the RMM driver which computes first computes anomalies of outgoing longwave raidation, 850 hPa wind and 200 hPa wind. Then, it regrids the data to 15S to 15N. Next, RMM is computed and a phase diagram, time series, and EOF plot are created. Inputs to the RMM driver include netCDF files that are in MET's netCDF version. In addition, a text file containing the listing of these input netCDF files for OLR, u850 and u200 is required. Some optional pre-processing steps include using pcp_combine to compute daily means and the mean daily annual cycle for the data.

METplus Workflow

The RMM driver script python code is run for each lead time on the forecast and observations data. This example loops by valid time for the model pre-processing, and valid time for the other steps. This version is set to only process the creation of anomalies, regridding, and RMM calculation, omitting the caluclation of daily means and the mean daily annucal cycle pre-processing steps. However, the configurations for pre-processing are available for user reference.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_RMM.conf. The file UserScript_obsERA_obsOnly_RMM/RMM_driver.py runs the python program and UserScript_obsERA_obsOnly_RMM.conf sets the variables for all steps of the RMM use case.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_RMM.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

# All steps, including creating daily means and mean daily annual cycle
#PROCESS_LIST = PcpCombine(mean_daily_annual_cycle_obs_wind), PcpCombine(mean_daily_annual_
↪cycle_obs_olr), PcpCombine(daily_mean_obs_wind), PcpCombine(daily_mean_obs_olr),␣
↪UserScript(create_mda_filelist), UserScript(harmonic_anomalies_olr), UserScript(harmonic_
↪anomalies_u850), UserScript(harmonic_anomalies_u200), RegridDataPlane(regrid_obs_olr),␣
↪RegridDataPlane(regrid_obs_u850), RegridDataPlane(regrid_obs_u200), UserScript(script_rmm)
# Computing anomalies, regridding, and RMM Analysis script

PROCESS_LIST = UserScript(create_mda_filelist), UserScript(harmonic_anomalies_olr),␣
↪UserScript(harmonic_anomalies_u850),  UserScript(harmonic_anomalies_u200),␣
↪RegridDataPlane(regrid_obs_olr), RegridDataPlane(regrid_obs_u850), RegridDataPlane(regrid_
↪obs_u200), UserScript(script_rmm)


###
# Time Info
```

(continues on next page)

(continued from previous page)

```
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2000010100
VALID_END = 2002123000
VALID_INCREMENT = 86400

LEAD_SEQ = 0

# variables referenced in other sections

# Run the obs for these cases
OBS_RUN = True
FCST_RUN = False


###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

REGRID_DATA_PLANE_VERIF_GRID = latlon 144 13 -15 0 2.5 2.5
REGRID_DATA_PLANE_METHOD = NEAREST
REGRID_DATA_PLANE_WIDTH = 1



###
# PCPCombine(mean_daily_annual_cycle_obs_wind) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Configurations for creating U200 and U850 mean daily annual cycle obs
# Mean daily annual cycle anomalies are computed for 1979 - 2001
[mean_daily_annual_cycle_obs_wind]

LOOP_BY = VALID
```

(continues on next page)

(continued from previous page)

```
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2012010100
VALID_END = 2012123100
VALID_INCREMENT = 86400

OBS_PCP_COMBINE_RUN = {OBS_RUN}

OBS_PCP_COMBINE_METHOD = USER_DEFINED

OBS_PCP_COMBINE_COMMAND = -derive mean {OBS_PCP_COMBINE_INPUT_DIR}/{OBS_PCP_COMBINE_INPUT_
↪TEMPLATE} -field 'name="U_P850_mean"; level="(*,*)"; set_attr_valid = "{valid?fmt=%Y%m%d_%H
↪%M%S}";' -field 'name="U_P200_mean"; level="(*,*)"; set_attr_valid = "{valid?fmt=%Y%m%d_%H
↪%M%S}";' -name U_P850_mean,U_P200_mean

OBS_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_
↪obsOnly_RMM/ERA/daily_mean
OBS_PCP_COMBINE_INPUT_TEMPLATE = ERA_wind_daily_mean_*{valid?fmt=%m%d}.nc

OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/mean_
↪daily_annual_cycle
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = ERA_wind_daily_annual_{valid?fmt=%m%d}.nc


###
# PCPCombine(mean_daily_annual_cycle_obs_olr) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Configurations for creating OLR mean daily annual cycle obs
# Mean daily annual cycle anomalies are computed for 1979 - 2001
[mean_daily_annual_cycle_obs_olr]

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2012010100
VALID_END = 2012123100
VALID_INCREMENT = 86400

OBS_PCP_COMBINE_RUN = {OBS_RUN}

OBS_PCP_COMBINE_METHOD = USER_DEFINED

OBS_PCP_COMBINE_COMMAND = -derive mean {OBS_PCP_COMBINE_INPUT_DIR}/{OBS_PCP_COMBINE_INPUT_
↪TEMPLATE} -field 'name="olr"; level="(*,*)";'
```

(continues on next page)

(continued from previous page)

```
OBS_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_
↪obsOnly_RMM/ERA/daily_mean
OBS_PCP_COMBINE_INPUT_TEMPLATE = ERA_OLR_daily_mean_*{valid?fmt=%m%d}.nc

OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/mean_
↪daily_annual_cycle
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = ERA_OLR_daily_annual_{valid?fmt=%m%d}.nc


###
# PCPCombine(daily_mean_obs_wind) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Configurations for creating U200 and U850 daily mean obs
[daily_mean_obs_wind]

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 1979010100
VALID_END = 2002123100
VALID_INCREMENT = 86400

OBS_PCP_COMBINE_RUN = {OBS_RUN}

OBS_PCP_COMBINE_METHOD = USER_DEFINED

OBS_PCP_COMBINE_COMMAND = -derive mean {OBS_PCP_COMBINE_INPUT_DIR}/{OBS_PCP_COMBINE_INPUT_
↪TEMPLATE} -field 'name="U"; level="P850"; set_attr_valid = "{valid?fmt=%Y%m%d_%H%M%S}";' -
↪field 'name="U"; level="P200"; set_attr_valid = "{valid?fmt=%Y%m%d_%H%M%S}";'

OBS_PCP_COMBINE_INPUT_DIR = /gpfs/fs1/collections/rda/data/ds627.0/ei.oper.an.pl
OBS_PCP_COMBINE_INPUT_TEMPLATE = {valid?fmt=%Y%m}/ei.oper.an.pl.regn128uv.{valid?fmt=%Y%m%d}*

OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/daily_
↪mean
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = ERA_wind_daily_mean_{valid?fmt=%Y%m%d}.nc


###
# PCPCombine(daily_mean_obs_olr) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

# Configurations for creating mean daily annual cycle obs OLR
```

(continues on next page)

(continued from previous page)

```
[daily_mean_obs_olr]

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 1979010100
VALID_END = 2002123100
VALID_INCREMENT = 86400

OBS_PCP_COMBINE_RUN = {OBS_RUN}

OBS_PCP_COMBINE_METHOD = USER_DEFINED

OBS_PCP_COMBINE_COMMAND = -add {OBS_PCP_COMBINE_INPUT_DIR}/{OBS_PCP_COMBINE_INPUT_TEMPLATE} -
↪field 'name="olr"; level="({valid?fmt=%Y%m%d_%H%M%S},*,*)"; file_type=NETCDF_NCCF;'

OBS_PCP_COMBINE_INPUT_DIR = /glade/u/home/kalb/MJO
OBS_PCP_COMBINE_INPUT_TEMPLATE = olr.1x.7920.nc

OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/daily_
↪mean
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = ERA_OLR_daily_mean_{valid?fmt=%Y%m%d}.nc


###
# UserScript(create_mda_filelist) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Creating a file list of the mean daily annual cycle files
# This is run separately since it has different start/end times
[create_mda_filelist]

# Find the files for each lead time
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

# Valid Begin and End Times for the CBL File Climatology
VALID_BEG = 2012010100
VALID_END = 2012123100
VALID_INCREMENT = 86400
LEAD_SEQ = 0

# Template of filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_
↪obsOnly_RMM/ERA/mean_daily_annual_cycle/ERA_OLR_daily_annual_{valid?fmt=%m%d}.nc,{INPUT_
↪BASE}/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/mean_daily_annual_cycle/
```

(continues on next page)

(continued from previous page)

```
↪ERA_wind_daily_annual_{valid?fmt=%m%d}.nc

# Name of the file containing the listing of input files
USER_SCRIPT_INPUT_TEMPLATE_LABELS = input_mean_daily_annual_infiles_olr,input_mean_daily_
↪annual_infiles_wind

# Placeholder command just to build the file list
# This just states that it's building the file list
USER_SCRIPT_COMMAND = echo Populated file list for Mean daily annual cycle Input


###
# UserScript(harmonic_anomalies_olr) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Configurations to create anomalies for OLR
[harmonic_anomalies_olr]

# list of strings to loop over for each run time.
# Run the user script once per lead
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

# Template of filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_
↪obsOnly_RMM/ERA/daily_mean/ERA_OLR_daily_mean_{valid?fmt=%Y%m%d}.nc

# Name of the file containing the listing of input files
# The options are OBS_OLR_INPUT, OBS_U850_INPUT, OBS_U200_INPUT, FCST_OLR_INPUT, FCST_U850_
↪INPUT, and FCST_U200_INPUT
# *** Make sure the order is the same as the order of templates listed in USER_SCRIPT_INPUT_
↪TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS = input_daily_mean_infiles

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_RMM/compute_harmonic_anomalies.py 'METPLUS_FILELIST_INPUT_MEAN_DAILY_ANNUAL_
↪INFILES_OLR' 'olr' 'olr_NA_mean' '{OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/
↪Anomaly' 'ERA_OLR_anom'


###
# UserScript(harmonic_anomalies_u850) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###
```

(continues on next page)

(continued from previous page)

```
# Configurations to create anomalies for U850
[harmonic_anomalies_u850]

# list of strings to loop over for each run time.
# Run the user script once per lead
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

# Template of filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_
↪obsOnly_RMM/ERA/daily_mean/ERA_wind_daily_mean_{valid?fmt=%Y%m%d}.nc

# Name of the file containing the listing of input files
# The options are OBS_OLR_INPUT, OBS_U850_INPUT, OBS_U200_INPUT, FCST_OLR_INPUT, FCST_U850_
↪INPUT, and FCST_U200_INPUT
# *** Make sure the order is the same as the order of templates listed in USER_SCRIPT_INPUT_
↪TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS = input_daily_mean_infiles

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_RMM/compute_harmonic_anomalies.py 'METPLUS_FILELIST_INPUT_MEAN_DAILY_ANNUAL_
↪INFILES_WIND' 'U_P850_mean' 'U_P850_mean' '{OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_
↪RMM/ERA/Anomaly' 'ERA_U850_anom'


###
# UserScript(harmonic_anomalies_u200) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Configurations to create anomalies for U200
[harmonic_anomalies_u200]

# list of strings to loop over for each run time.
# Run the user script once per lead
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

# Template of filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_
↪obsOnly_RMM/ERA/daily_mean/ERA_wind_daily_mean_{valid?fmt=%Y%m%d}.nc

# Name of the file containing the listing of input files
# The options are OBS_OLR_INPUT, OBS_U850_INPUT, OBS_U200_INPUT, FCST_OLR_INPUT, FCST_U850_
↪INPUT, and FCST_U200_INPUT
```

(continues on next page)

(continued from previous page)

```
# *** Make sure the order is the same as the order of templates listed in USER_SCRIPT_INPUT_
↪TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS = input_daily_mean_infiles

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_RMM/compute_harmonic_anomalies.py 'METPLUS_FILELIST_INPUT_MEAN_DAILY_ANNUAL_
↪INFILES_WIND' 'U_P200_mean' 'U_P200_mean' '{OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_
↪RMM/ERA/Anomaly' 'ERA_U200_anom'


###
# RegridDataPlane(regrid_obs_olr) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Configurations for regrid_data_plane: Regrid OLR to -15 to 15 latitude
[regrid_obs_olr]

OBS_REGRID_DATA_PLANE_RUN = {OBS_RUN}

REGRID_DATA_PLANE_ONCE_PER_FIELD = False

OBS_REGRID_DATA_PLANE_VAR1_NAME = olr_anom
OBS_REGRID_DATA_PLANE_VAR1_LEVELS = "(*,*)"
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = OLR_anom

OBS_REGRID_DATA_PLANE_INPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/
↪Anomaly
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/
↪Regrid

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = ERA_OLR_anom_{lead?fmt=%H%M%S}L_{valid?fmt=%Y%m%d}_
↪{valid?fmt=%H%M%S}V.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = ERA_OLR_{valid?fmt=%Y%m%d}.nc


###
# RegridDataPlane(regrid_obs_u850) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Configurations for regrid_data_plane: Regrid u850 to -15 to 15 latitude
[regrid_obs_u850]
```

(continues on next page)

(continued from previous page)

```
OBS_REGRID_DATA_PLANE_RUN = {OBS_RUN}

REGRID_DATA_PLANE_ONCE_PER_FIELD = False

OBS_REGRID_DATA_PLANE_VAR1_NAME = U_P850_mean_anom
OBS_REGRID_DATA_PLANE_VAR1_LEVELS = "(*,*)"
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = U_P850_anom

OBS_REGRID_DATA_PLANE_INPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/
↪Anomaly
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/
↪Regrid

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = ERA_U850_anom_{lead?fmt=%H%M%S}L_{valid?fmt=%Y%m%d}_
↪{valid?fmt=%H%M%S}V.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = ERA_U850_{valid?fmt=%Y%m%d}.nc


###
# RegridDataPlane(regrid_obs_u200) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

# Configurations for regrid_data_plane: Regrid u200 to -15 to 15 latitude
[regrid_obs_u200]

OBS_REGRID_DATA_PLANE_RUN = {OBS_RUN}

REGRID_DATA_PLANE_ONCE_PER_FIELD = False

OBS_REGRID_DATA_PLANE_VAR1_NAME = U_P200_mean_anom
OBS_REGRID_DATA_PLANE_VAR1_LEVELS = "(*,*)"
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = U_P200_anom

OBS_REGRID_DATA_PLANE_INPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/
↪Anomaly
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/
↪Regrid

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = ERA_U200_anom_{lead?fmt=%H%M%S}L_{valid?fmt=%Y%m%d}_
↪{valid?fmt=%H%M%S}V.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = ERA_U200_{valid?fmt=%Y%m%d}.nc


# Configurations for the RMM analysis script
```

(continues on next page)

(continued from previous page)

```
[user_env_vars]
# Whether to Run the model or obs
RUN_OBS = {OBS_RUN}
RUN_FCST = {FCST_RUN}

# Make OUTPUT_BASE Available to the script
SCRIPT_OUTPUT_BASE = {OUTPUT_BASE}

# Number of obs per day
OBS_PER_DAY = 1

# Variable names for OLR, U850, U200
OBS_OLR_VAR_NAME = OLR_anom
OBS_U850_VAR_NAME = U_P850_anom
OBS_U200_VAR_NAME = U_P200_anom

# EOF Filename
OLR_EOF_INPUT_TEXTFILE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_
↪RMM/EOF/rmm_olr_eofs.txt
U850_EOF_INPUT_TEXTFILE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_
↪RMM/EOF/rmm_u850_eofs.txt
U200_EOF_INPUT_TEXTFILE = {INPUT_BASE}/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_
↪RMM/EOF/rmm_u200_eofs.txt

# Normalization factors for RMM
RMM_OLR_NORM = 15.11623
RMM_U850_NORM = 1.81355
RMM_U200_NORM = 4.80978
PC1_NORM = 8.618352504159244
PC2_NORM = 8.40736449709697

# Output Directory for the plots
# If not set, it this will default to {OUTPUT_BASE}/plots
RMM_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/plots

# EOF plot information
EOF_PLOT_OUTPUT_NAME = RMM_EOFs
EOF_PLOT_OUTPUT_FORMAT = png

# Phase Plot start date, end date, output name, and format
PHASE_PLOT_TIME_BEG = 2002010100
PHASE_PLOT_TIME_END = 2002123000
PHASE_PLOT_TIME_FMT = {VALID_TIME_FMT}
OBS_PHASE_PLOT_OUTPUT_NAME = obs_RMM_comp_phase
OBS_PHASE_PLOT_OUTPUT_FORMAT = png
```

(continues on next page)

(continued from previous page)

```
# Time Series Plot start date, end date, output name, and format
TIMESERIES_PLOT_TIME_BEG = 2002010100
TIMESERIES_PLOT_TIME_END = 2002123000
TIMESERIES_PLOT_TIME_FMT = {VALID_TIME_FMT}
OBS_TIMESERIES_PLOT_OUTPUT_NAME = obs_RMM_time_series
OBS_TIMESERIES_PLOT_OUTPUT_FORMAT = png


###
# UserScript(script_rmm) Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

# Configurations for UserScript: Run the RMM Analysis driver
[script_rmm]
# list of strings to loop over for each run time.
# Run the user script once per lead
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

# Template of filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/Regrid/
 ↪ERA_OLR_{valid?fmt=%Y%m%d}.nc,{OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/ERA/
 ↪Regrid/ERA_U850_{valid?fmt=%Y%m%d}.nc,{OUTPUT_BASE}/s2s_mjo/UserScript_obsERA_obsOnly_RMM/
 ↪ERA/Regrid/ERA_U200_{valid?fmt=%Y%m%d}.nc

# Name of the file containing the listing of input files
# The options are OBS_OLR_INPUT, OBS_U850_INPUT, OBS_U200_INPUT, FCST_OLR_INPUT, FCST_U850_
 ↪INPUT, and FCST_U200_INPUT
# *** Make sure the order is the same as the order of templates listed in USER_SCRIPT_INPUT_
 ↪TEMPLATE
USER_SCRIPT_INPUT_TEMPLATE_LABELS = OBS_OLR_INPUT,OBS_U850_INPUT, OBS_U200_INPUT

# Command to run the user script with input configuration file
USER_SCRIPT_COMMAND = {METPLUS_BASE}/parm/use_cases/model_applications/s2s_mjo/UserScript_
 ↪obsERA_obsOnly_RMM/RMM_driver.py
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

Python Scripts

The RMM driver script orchestrates the calculation of the MJO indices and the generation of three RMM plots: parm/use_cases/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_RMM/RMM_driver.py: The harmonic anomalies script creates anomalies of input data using a harmonic analysis: parm/use_cases/model_applications/s2s_mjo/UserScript_obsERA_obsOnly_RMM/compute_harmonic_anomalies.py

```
#!/usr/bin/env python3
"""
Driver Script to Compute RMM index from input U850, U200 and OLR data. Data is averaged from␣
↪20S-20N
"""

import numpy as np
import xarray as xr
import pandas as pd
import datetime
import glob
import os
import sys
import warnings

import metcalcpy.contributed.rmm_omi.compute_mjo_indices as cmi
import metplotpy.contributed.mjo_rmm_omi.plot_mjo_indices as pmi
import METreadnc.util.read_netcdf as read_netcdf


def read_rmm_eofs(olrfile, u850file, u200file):
    """
    Read the OMI EOFs from file and into a xarray DataArray.
    :param eofpath: filepath to the location of the eof files
    :return: EOF1 and EOF2 2D DataArrays
    """

    # observed EOFs from BOM Australia are saved in individual text files for each variable
    # horizontal resolution of EOFs is 2.5 degree and longitudes go from 0 - 375.5, column1␣
```

(continues on next page)

(continued from previous page)

```
↪is eof1
    # column 2 is eof2 in each file
    EOF1 = xr.DataArray(np.empty([3,144]),dims=['var','lon'],
    coords={'var':['olr','u850','u200'], 'lon':np.arange(0,360,2.5)})
    EOF2 = xr.DataArray(np.empty([3,144]),dims=['var','lon'],
    coords={'var':['olr','u850','u200'], 'lon':np.arange(0,360,2.5)})
    nlon = len(EOF1['lon'])

    tmp = pd.read_csv(olrfile, header=None, delim_whitespace=True, names=['eof1','eof2'])
    EOF1[0,:] = tmp.eof1.values
    EOF2[0,:] = tmp.eof2.values
    tmp = pd.read_csv(u850file, header=None, delim_whitespace=True, names=['eof1','eof2'])
    EOF1[1,:] = tmp.eof1.values
    EOF2[1,:] = tmp.eof2.values
    tmp = pd.read_csv(u200file, header=None, delim_whitespace=True, names=['eof1','eof2'])
    EOF1[2,:] = tmp.eof1.values
    EOF2[2,:] = tmp.eof2.values

    return EOF1, EOF2


def run_rmm_steps(inlabel, spd, EOF1, EOF2, oplot_dir):

    # Get OLR, U850, U200 file listings and variable names
    olr_filetxt = os.environ['METPLUS_FILELIST_'+inlabel+'_OLR_INPUT']
    u850_filetxt = os.environ['METPLUS_FILELIST_'+inlabel+'_U850_INPUT']
    u200_filetxt = os.environ['METPLUS_FILELIST_'+inlabel+'_U200_INPUT']

    olr_var = os.environ[inlabel+'_OLR_VAR_NAME']
    u850_var = os.environ[inlabel+'_U850_VAR_NAME']
    u200_var = os.environ[inlabel+'_U200_VAR_NAME']

    # Read the listing of OLR, U850, U200 files
    with open(olr_filetxt) as ol:
        olr_input_files = ol.read().splitlines()
    if (olr_input_files[0] == 'file_list'):
            olr_input_files = olr_input_files[1:]
    with open(u850_filetxt) as u8:
        u850_input_files = u8.read().splitlines()
    if (u850_input_files[0] == 'file_list'):
            u850_input_files = u850_input_files[1:]
    with open(u200_filetxt) as u2:
        u200_input_files = u2.read().splitlines()
    if (u200_input_files[0] == 'file_list'):
            u200_input_files = u200_input_files[1:]
```

(continues on next page)

(continued from previous page)

```
# Check the input data to make sure it's not all missing
olr_allmissing = all(elem == 'missing' for elem in olr_input_files)
if olr_allmissing:
    raise IOError ('No input OLR files were found, check file paths')
u850_allmissing = all(elem == 'missing' for elem in u850_input_files)
if u850_allmissing:
    raise IOError('No input U850 files were found, check file paths')
u200_allmissing = all(elem == 'missing' for elem in u200_input_files)
if u200_allmissing:
    raise IOError('No input U200 files were found, check file paths')


# Read OLR, U850, U200 data from file
netcdf_reader_olr = read_netcdf.ReadNetCDF()
ds_olr = netcdf_reader_olr.read_into_xarray(olr_input_files)

netcdf_reader_u850 = read_netcdf.ReadNetCDF()
ds_u850 = netcdf_reader_u850.read_into_xarray(u850_input_files)

netcdf_reader_u200 = read_netcdf.ReadNetCDF()
ds_u200 = netcdf_reader_u200.read_into_xarray(u200_input_files)


time = []
for din in range(len(ds_olr)):
    colr = ds_olr[din]
    ctime =  datetime.datetime.strptime(colr[olr_var].valid_time,'%Y%m%d_%H%M%S')
    time.append(ctime.strftime('%Y-%m-%d'))
    colr = colr.assign_coords(time=ctime)
    ds_olr[din] = colr.expand_dims("time")

    cu850 = ds_u850[din]
    cu850 = cu850.assign_coords(time=ctime)
    ds_u850[din] = cu850.expand_dims("time")

    cu200 = ds_u200[din]
    cu200 = cu200.assign_coords(time=ctime)
    ds_u200[din] = cu200.expand_dims("time")

time = np.array(time,dtype='datetime64[D]')

everything_olr = xr.concat(ds_olr,"time")
olr = everything_olr[olr_var]
olr = olr.mean('lat')
```

(continues on next page)

(continued from previous page)

```
    print(olr.min(), olr.max())

    everything_u850 = xr.concat(ds_u850,"time")
    u850 = everything_u850[u850_var]
    u850 = u850.mean('lat')
    print(u850.min(), u850.max())

    everything_u200 = xr.concat(ds_u200,"time")
    u200 = everything_u200[u200_var]
    u200 = u200.mean('lat')
    print(u200.min(), u200.max())


    # Get normalization factors for use
    rmm_norm = [float(os.environ['RMM_OLR_NORM']),float(os.environ['RMM_U850_NORM']),
↪float(os.environ['RMM_U200_NORM'])]
    pc_norm = [float(os.environ['PC1_NORM']),float(os.environ['PC2_NORM'])]

    # project data onto EOFs
    PC1, PC2 = cmi.rmm(olr, u850, u200, time, spd, EOF1, EOF2, rmm_norm, pc_norm)
    print(PC1.min(), PC1.max())

    # Get times for the PC phase diagram
    plase_plot_time_format = os.environ['PHASE_PLOT_TIME_FMT']
    phase_plot_start_time = datetime.datetime.strptime(os.environ['PHASE_PLOT_TIME_BEG'],
↪plase_plot_time_format)
    phase_plot_end_time = datetime.datetime.strptime(os.environ['PHASE_PLOT_TIME_END'],plase_
↪plot_time_format)
    PC1_pcplot = PC1.sel(time=slice(phase_plot_start_time,phase_plot_end_time))
    PC2_pcplot = PC2.sel(time=slice(phase_plot_start_time,phase_plot_end_time))
    pc_ntim_plot = len(PC1_pcplot)
    PC1_pcplot = PC1_pcplot[0:pc_ntim_plot]
    PC2_pcplot = PC2_pcplot[0:pc_ntim_plot]

    # Get the output name and format for the PC plase diagram
    phase_plot_name = os.path.join(oplot_dir,os.environ.get(inlabel+'_PHASE_PLOT_OUTPUT_NAME
↪',inlabel+'_RMM_comp_phase'))
    phase_plot_format = os.environ.get(inlabel+'_PHASE_PLOT_OUTPUT_FORMAT','png')

    # plot the PC phase diagram
    pmi.phase_diagram('RMM',PC1_pcplot,PC2_pcplot,np.array(PC1_pcplot['time'].dt.strftime("
↪%Y-%m-%d").values),
        np.ndarray.tolist(PC1_pcplot['time.month'].values),np.ndarray.tolist(PC1_pcplot[
↪'time.day'].values),
        phase_plot_name, phase_plot_format)
```

(continues on next page)

(continued from previous page)

```
    # Get times for the PC time series plot
    timeseries_plot_time_format = os.environ['TIMESERIES_PLOT_TIME_FMT']
    timeseries_plot_start_time = datetime.datetime.strptime(os.environ['TIMESERIES_PLOT_TIME_
↪BEG'],plase_plot_time_format)
    timeseries_plot_end_time = datetime.datetime.strptime(os.environ['TIMESERIES_PLOT_TIME_
↪END'],plase_plot_time_format)
    PC1_tsplot = PC1.sel(time=slice(timeseries_plot_start_time,timeseries_plot_end_time))
    PC2_tsplot = PC2.sel(time=slice(timeseries_plot_start_time,timeseries_plot_end_time))
    ts_ntim_plot = len(PC1_tsplot)
    PC1_tsplot = PC1_tsplot[0:ts_ntim_plot]
    PC2_tsplot = PC2_tsplot[0:ts_ntim_plot]

    # Get the ouitput name and format for the PC Timeseries plot
    timeseries_plot_name = os.path.join(oplot_dir,os.environ.get(inlabel+'_TIMESERIES_PLOT_
↪OUTPUT_NAME',
        inlabel+'_RMM_timeseries'))
    timeseries_plot_format = os.environ.get(inlabel+'_TIMESERIES_PLOT_OUTPUT_FORMAT','png')

    # plot PC time series
    pmi.pc_time_series('RMM',PC1_tsplot,PC2_tsplot,np.array(PC1_tsplot['time'].values),
        np.ndarray.tolist(PC1_tsplot['time.month'].values),np.ndarray.tolist(PC1_tsplot[
↪'time.day'].values),
        timeseries_plot_name,timeseries_plot_format)


def main():

    # Get the EOF files and EOF plot variables
    olr_eoffile = os.environ['OLR_EOF_INPUT_TEXTFILE']
    u850_eoffile = os.environ['U850_EOF_INPUT_TEXTFILE']
    u200_eoffile = os.environ['U200_EOF_INPUT_TEXTFILE']

    # Get Number of Obs per day
    spd = os.environ.get('OBS_PER_DAY',1)

    # Check for an output plot directory
    oplot_dir = os.environ.get('RMM_PLOT_OUTPUT_DIR','')
    if not oplot_dir:
        obase = os.environ['SCRIPT_OUTPUT_BASE']
        oplot_dir = os.path.join(obase,'plots')
    if not os.path.exists(oplot_dir):
        os.makedirs(oplot_dir)

    # Read in the EOFs and plot and get plot name and format
```

(continues on next page)

(continued from previous page)

```
    EOF1, EOF2 = read_rmm_eofs(olr_eoffile, u850_eoffile, u200_eoffile)
    eof_plot_name = os.path.join(oplot_dir,os.environ.get('EOF_PLOT_OUTPUT_NAME','RMM_EOFs'))
    eof_plot_format = os.environ.get('EOF_PLOT_OUTPUT_FORMAT','png')
    pmi.plot_rmm_eofs(EOF1, EOF2, eof_plot_name, eof_plot_format)

    #  Determine if doing forecast or obs
    run_obs_rmm = os.environ.get('RUN_OBS', 'False').lower()
    run_fcst_rmm = os.environ.get('RUN_FCST', 'False').lower()

    if (run_obs_rmm == 'true'):
        run_rmm_steps('OBS', spd, EOF1, EOF2, oplot_dir)

    if (run_fcst_rmm == 'true'):
        run_rmm_steps('FCST', spd, EOF1, EOF2, oplot_dir)

    # nothing selected
    if (run_obs_rmm == 'false') and (run_fcst_rmm == 'false'):
        warnings.warn('Forecast and Obs runs not selected, nothing will be calculated')
        warnings.warn('Set RUN_FCST or RUN_OBS in the [user_en_vars] section to generate␣
↪output')


if __name__ == "__main__":
    main()
```

```
#!/usr/bin/env python3
import numpy as np
import xarray as xr
import glob
import os
import sys
import datetime
import METreadnc.util.read_netcdf as read_netcdf

input_mean_daily_annual_infiles_list = os.environ[sys.argv[1]]
dm_var = sys.argv[2]
mda_var = sys.argv[3]
anom_output_dir = sys.argv[4]
anom_output_base = sys.argv[5]
input_daily_mean_infiles_list = os.environ['METPLUS_FILELIST_INPUT_DAILY_MEAN_INFILES']

# Environment variables for script
nobs = int(os.environ.get('OBS_PER_DAY',1))
out_var = dm_var+'_anom'
```

(continues on next page)

(continued from previous page)

```
# Read the listing of files
with open(input_daily_mean_infiles_list) as idm:
    input_daily_mean_infiles = idm.read().splitlines()
if (input_daily_mean_infiles[0] == 'file_list'):
    input_daily_mean_infiles = input_daily_mean_infiles[1:]

with open(input_mean_daily_annual_infiles_list) as imda:
    input_mean_daily_annual_infiles = imda.read().splitlines()
if (input_mean_daily_annual_infiles[0] == 'file_list'):
    input_mean_daily_annual_infiles = input_mean_daily_annual_infiles[1:]


# Read in the data
netcdf_reader = read_netcdf.ReadNetCDF()
dm_orig = netcdf_reader.read_into_xarray(input_daily_mean_infiles)
# Add some needed attributes
dm_list = []
time_dm = []
yr_dm = []
doy_dm = []
for din in dm_orig:
    ctime =  datetime.datetime.strptime(din[dm_var].valid_time,'%Y%m%d_%H%M%S')
    time_dm.append(ctime.strftime('%Y-%m-%d'))
    yr_dm.append(int(ctime.strftime('%Y')))
    doy_dm.append(int(ctime.strftime('%j')))
    din = din.assign_coords(time=ctime)
    din = din.expand_dims("time")
    dm_list.append(din)
time_dm = np.array(time_dm,dtype='datetime64[D]')
yr_dm = np.array(yr_dm)
doy_dm = np.array(doy_dm)
everything = xr.concat(dm_list,"time")
dm_data = np.array(everything[dm_var])

netcdf_reader2 = read_netcdf.ReadNetCDF()
mda_orig = netcdf_reader2.read_into_xarray(input_mean_daily_annual_infiles)
# Add some needed attributes
mda_list = []
time_mda = []
for din in mda_orig:
    ctime =  datetime.datetime.strptime(din[mda_var].valid_time,'%Y%m%d_%H%M%S')
    time_mda.append(ctime.strftime('%Y-%m-%d'))
    din = din.assign_coords(time=ctime)
    din = din.expand_dims("time")
    mda_list.append(din)
```

(continues on next page)

(continued from previous page)

```
time_mda = np.array(time_mda,dtype='datetime64[D]')
everything2 = xr.concat(mda_list,"time")
mda_data = np.array(everything2[mda_var])

# Harmonic Analysis, first step is Forward Fast Fourier Transform
clmfft =  np.fft.rfft(mda_data,axis=0)

smthfft = np.zeros(clmfft.shape,dtype=complex)
for f in np.arange(0,3):
    smthfft[f,:,:] = clmfft[f,:,:]

clmout = np.fft.irfft(smthfft,axis=0)

# Subtract the clmout from the data to create anomalies, each year at a time
yrstrt = yr_dm[0]
yrend = yr_dm[-1]
anom = np.zeros(dm_data.shape)

for y in np.arange(yrstrt,yrend+1,1):
    curyr = np.where(yr_dm == y)
    dd = doy_dm[curyr] - 1
    ndd = len(curyr[0])
    clmshp = [np.arange(dd[0]*nobs,dd[0]*nobs+ndd,1)]
    anom[curyr,:,:] = dm_data[curyr,:,:] - clmout[clmshp,:,:]

# Assign to an xarray and write output
if not os.path.exists(anom_output_dir):
    os.makedirs(anom_output_dir)
for o in np.arange(0,len(dm_orig)):
    dm_orig_cur = dm_orig[o]
    dout = xr.Dataset({out_var: (("lat", "lon"),anom[o,:,:])},
    coords={"lat": dm_orig_cur.coords['lat'], "lon": dm_orig_cur.coords['lon']},
    attrs=dm_orig_cur.attrs)
    dout[out_var].attrs = dm_orig_cur[dm_var].attrs
    dout[out_var].attrs['long_name'] = dm_orig_cur[dm_var].attrs['long_name']+' Anomalies'
    dout[out_var].attrs['name'] = out_var

    # write to a file
    cvtime = datetime.datetime.strptime(dm_orig_cur[dm_var].valid_time,'%Y%m%d_%H%M%S')
    citime = datetime.datetime.strptime(dm_orig_cur[dm_var].init_time,'%Y%m%d_%H%M%S')
    cltime = (cvtime - citime)
    leadmin,leadsec = divmod(cltime.total_seconds(), 60)
    leadhr,leadmin = divmod(leadmin,60)
    lead_str = str(int(leadhr)).zfill(2)+str(int(leadmin)).zfill(2)+str(int(leadsec)).
↪zfill(2)
```

(continues on next page)

(continued from previous page)

```
    dout.to_netcdf(os.path.join(anom_output_dir,anom_output_base+'_'+lead_str+'L_'+cvtime.
↪strftime('%Y%m%d')+'_'+cvtime.strftime('%H%M%S')+'V.nc'))
```

Running METplus

This use case is run in the following ways:

1) Passing in UserScript_obsERA_obsOnly_RMM.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_RMM.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_obsERA_obsOnly_RMM.py:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s_mjo/UserScript_
↪obsERA_obsOnly_RMM.conf
```

The following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/s2s_mjo/UserScript_obsERA_obsOnly_RMM. This may include the regridded data and daily averaged files. In addition, three plots will be generated, a phase diagram, time series, and EOF plot, and the output location can be specified as RMM_PLOT_OUTPUT_DIR. If it is not specified, plots will be sent to model_applications/s2s_mjo/UserScript_obsERA_obsOnly_RMM/plots (relative to **OUTPUT_BASE**).

Keywords

Note:

- S2SAppUseCase
- S2SMJOAppUseCase
- NetCDFFileUseCase
- RegridDataPlaneUseCase
- PCPCombineUseCase
- METcalcpyUseCase
- METplotpyUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1997) to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s_mjo-UserScript_obsERA_obsOnly_RMM.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13 Short Range

High resolution model configurations (1-4km) usually producing forecasts between 0-3 days (also referred to as limited area models, stand-alone regional, and short range weather applications); Previously named Convection Allowing Models

7.2.16.13.1 UserScript: Physics Tendency Vertical Profile plot

model_applications/short_range/UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalProfile.conf

Scientific Objective

To plot tendencies of temperature, moisture, and wind components averaged over a time window and spatial domain. Tendencies are partitioned into physics parameterizations and dynamics. Physics parameterizations include schemes like deep convection, convective gravity wave drag, short wave radiation, plantetary boundary layer, microphysics, and others. Non-physics tendencies (or dynamics) are due to horizontal and vertical motion. The residual (which should be zero) is the difference between the actual change in the state variable over the requested time window and the expected change due to physics parameterizations and dynamics tendencies. One can plot a single tendency component at multiple pressure levels or plot all tendency components at a single pressure level. This use case illustrates how to generate the vertical profile plot. The METplotpy source code is needed to generate the plot. Clone the METplotpy repository (https://github.com/dtcenter/METplotpy) under the same base directory as the METPLUS_BASE

directory so that the METplus and METplotpy directories are under the same base directory (i.e. if the METPLUS_BASE directory is /home/username/working/METplus, then clone the METplotpy source code into the /home/username/working directory).

Datasets

- Forecast dataset: FV3 3-D history file with physics and dynamics tendencies
- Grid specification: FV3 2-D grid specification file with latitude and longitude of each grid point
- Mid-CONUS Shapefiles:
  - MID_CONUS.cpg
  - MID_CONUS.dbf
  - MID_CONUS.poly
  - MID_CONUS.prj
  - MID_CONUS.shp
  - MID_CONUS.shx

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1746) section for more information.

External Dependencies

You will need to use a versio of Python 3.86 that has the following packages installed:

- cartopy (0.20.3 only)
- matplotlib
- metpy
- numpy
- pandas
- shapely
- xarray

METplus Components

This use case runs the METplotpy vert_profile_fv3.py script to generate the vertical profile plot.

METplus Workflow

This use case does not loop but plots physics tendency data that has been subsetted to one date: 2019-05-04.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/short_range/UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalProfile.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/
↪UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalProfile.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###
```

(continues on next page)

(continued from previous page)

```
LOOP_BY = REALTIME
VALID_TIME_FMT = %Y
VALID_BEG = 2019

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

FV3_HISTORY_FILE = {INPUT_BASE}/model_applications/short_range/UserScript_fcstFV3_fcstOnly_
↪PhysicsTendency/fv3_history.nc
GRID_SPEC_FILE = {INPUT_BASE}/model_applications/short_range/UserScript_fcstFV3_fcstOnly_
↪PhysicsTendency/grid_spec.nc
USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/model_applications/short_range/UserScript_
↪fcstFV3_fcstOnly_PhysicsTendency_VerticalProfile/vertical_profile_plot.py {PARM_BASE}/use_
↪cases/model_applications/short_range/UserScript_fcstFV3_fcstOnly_PhysicsTendency_
↪VerticalProfile/physics_tendency_vertical_profile.yaml {FV3_HISTORY_FILE} {GRID_SPEC_FILE}␣
↪tmp -t 2 -v 20190504T14 -o {OUTPUT_BASE}/plots/short_range-physics_tendency_vertical_
↪profile.png -s {INPUT_BASE}/model_applications/short_range/UserScript_fcstFV3_fcstOnly_
↪PhysicsTendency/shapefiles/MID_CONUS --nofineprint


[user_env_vars]

# VerticalProfile plot specific variables

LOG_FILE = "VerticalProfile.log"

LOG_LEVEL = "DEBUG"

YAML_CONFIG_NAME = {PARM_BASE}/use_cases/model_applications/short_range/UserScript_fcstFV3_
↪fcstOnly_PhysicsTendency_VerticalProfile/physics_tendency_vertical_profile.yaml
```

MET Configuration

No MET tools are used in this use case.

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

Note: See the *MODE MET Configuration* (page 174) section of the User's Guide for more information on the environment variables used in the file below:

Running METplus

This use case can be run in the following way:

1) Passing in UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalProfile.conf then a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/short_range/
↪UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalProfile.conf  /path/to/user_
↪system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalProfile.conf:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/short_range/
↪UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalProfile.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. The following file will be created:

short_range-physics_tendency_vertical_profile.png

Keywords

Note:

- MediumRangeAppUseCase
- PhysicsTendency
- ValidationUseCase
- ShortRangeAppUseCase
- S2SAppUseCase
- METplotpyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalProfile.p

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.2 UserScript: Physics Tendency Planview Plot

model_applications/ short_range/ UserScript_fcstFV3_fcstOnly_PhysicsTendency_Planview.conf

Scientific Objective

To plot tendencies of temperature, moisture, and wind components averaged over a time window and spatial domain. Tendencies are partitioned into physics parameterizations and dynamics. Physics parameterizations include schemes like deep convection, convective gravity wave drag, short wave radiation, plantetary boundary layer, microphysics, and others. Non-physics tendencies (or dynamics) are due to horizontal and vertical motion. The residual (which should be zero) is the difference between the actual change in the state variable over the requested time window and the expected change due to physics parameterizations and dynamics tendencies. One can plot a single tendency component at multiple pressure levels or plot all tendency components at a single pressure level. This use case illustrates how to generate plan views (horizontal cross sections). The METplotpy source code is needed to generate the plot. Clone the METplotpy

repository (https://github.com/dtcenter/METplotpy) under the same base directory as the METPLUS_BASE directory so that the METplus and METplotpy directories are under the same base directory (i.e. if the METPLUS_BASE directory is /home/username/working/METplus, then clone the METplotpy source code into the /home/username/working directory).

Datasets

- Forecast dataset: FV3 3-D history file with physics and dynamics tendencies
- Grid specification: FV3 2-D grid specification file with latitude and longitude of each grid point

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See the "Running METplus" section below for more information.

External Dependencies

You will need to use a versio of Python 3.86 that has the following packages installed:

- cartopy (0.20.3 only)
- matplotlib
- metpy
- numpy
- pandas
- shapely
- xarray

METplus Components

This use case runs the METplotpy planview_fv3.py script to generate the plan views.

METplus Workflow

This use case does not loop but plots physics tendency data that has been subsetted to one date: 2019-05-04.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/short_range/UserScript_fcstFV3_fcstOnly_PhysicsTendency_Planview.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/
↪UserScript_fcstFV3_fcstOnly_PhysicsTendency_Planview.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = REALTIME
VALID_TIME_FMT = %Y
VALID_BEG = 2019

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

```

(continues on next page)

(continued from previous page)

```
FV3_HISTORY_FILE = {INPUT_BASE}/model_applications/short_range/UserScript_fcstFV3_fcstOnly_
↪PhysicsTendency/fv3_history.nc
GRID_SPEC_FILE = {INPUT_BASE}/model_applications/short_range/UserScript_fcstFV3_fcstOnly_
↪PhysicsTendency/grid_spec.nc
PRESSURE_LEVEL = 500
USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/model_applications/short_range/UserScript_
↪fcstFV3_fcstOnly_PhysicsTendency_Planview/planview_plot.py {PARM_BASE}/use_cases/model_
↪applications/short_range/UserScript_fcstFV3_fcstOnly_PhysicsTendency_Planview/physics_
↪tendency_planview.yaml {FV3_HISTORY_FILE} {GRID_SPEC_FILE} tmp pbl -p {PRESSURE_LEVEL} -t␣
↪1 -v 20190504T14 -o {OUTPUT_BASE}/plots/short_range-physics_tendency_planview.png --
↪nofineprint

[user_env_vars]

# Planview plot specific variables

LOG_FILE = "planview.log"

LOG_LEVEL = "DEBUG"

YAML_CONFIG_NAME = {PARM_BASE}/use_cases/model_applications/short_range/UserScript_fcstFV3_
↪fcstOnly_PhysicsTendency_Planview/physics_tendency_planview.yaml
```

MET Configuration

No MET tools are used in this use case.

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

Running METplus

This use case can be run in the following way:

1) Passing in UserScript_fcstFV3_fcstOnly_PhysicsTendency_Planview.conf then a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/short_range/UserScript_
↪fcstFV3_fcstOnly_PhysicsTendency_Planview.conf  /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_fcstFV3_fcstOnly_PhysicsTendency_Planview.conf:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/short_range/UserScript_
↪fcstFV3_fcstOnly_PhysicsTendency_Planview.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. The following file will be created:

physics_tendency_planview.png

Keywords

Note:

- MediumRangeAppUseCase
- PhysicsTendency
- ValidationUseCase
- ShortRangeAppUseCase
- S2SAppUseCase
- METplotpyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-UserScript_fcstFV3_fcstOnly_PhysicsTendency_Planview.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.3 Grid-Stat: Surrogate Severe and Practically Perfect Evaluation

model_applications/ short_range/ GridStat_fcstHRRR_obsPracPerfect _SurrogateSevere.conf

Scientific Objective

To evaluate the surrogate severe forecasts at predicting Severe weather using the (12Z - 12Z) practically perfect storm reports.

Datasets

- Forecast dataset: HRRR Surrogate Severe Data
- Observation dataset: Practically Perfect from Local Storm Reports.

METplus Components

This use case runs grid_stat to create categorical statistics for Surrogate Severe derived from the HRRR model and Practially Perfect Analysis derived from local storm reports.

METplus Workflow

The grid_stat tool is run for each time. This example loops by valid time. It processes 1 valid time, listed below.

Valid: 2020-02-06_12Z
Forecast lead: 36

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/short_range/GridStat_fcstHRRR_obsPracPerfect_SurrogateSevere.conf

```
[config]

# Documentation for this use case can be found at
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/GridStat_
↪fcstHRRR_obsPracPerfect_SurrogateSevere.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG=2020020612
VALID_END=2020020612
VALID_INCREMENT=86400

INIT_SEQ = 0
LEAD_SEQ_MIN = 36
LEAD_SEQ_MAX = 36


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/surrogate_severe_prac_
↪perfect
```

(continues on next page)

(continued from previous page)

```
FCST_GRID_STAT_INPUT_TEMPLATE = surrogate_severe_{init?fmt=%Y%m%d}_{lead?fmt=%HHH}V_regrid.nc

OBS_GRID_STAT_INPUT_DIR = {FCST_GRID_STAT_INPUT_DIR}
OBS_GRID_STAT_INPUT_TEMPLATE = StormReps_211_Probs.{init?fmt=%Y%m%d}.nc

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/short_range/surrogate_severe_prac_
↪perfect/grid_stat


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = HRRR
OBTYPE = PP

FCST_VAR1_NAME = MXUPHL_prob_75
FCST_VAR1_LEVELS = "(*,*)"
FCST_VAR1_THRESH = ge0.02, ge0.05, ge0.10, ge0.10, ge0.15, ge0.30, ge0.45, ge0.60

FCST_VAR2_NAME = MXUPHL_prob_80
FCST_VAR2_LEVELS = {FCST_VAR1_LEVELS}
FCST_VAR2_THRESH = {FCST_VAR1_THRESH}

FCST_VAR3_NAME = MXUPHL_prob_85
FCST_VAR3_LEVELS = {FCST_VAR1_LEVELS}
FCST_VAR3_THRESH = {FCST_VAR1_THRESH}

FCST_VAR4_NAME = MXUPHL_prob_90
FCST_VAR4_LEVELS = {FCST_VAR1_LEVELS}
FCST_VAR4_THRESH = {FCST_VAR1_THRESH}

FCST_VAR5_NAME = MXUPHL_prob_95
FCST_VAR5_LEVELS = {FCST_VAR1_LEVELS}
FCST_VAR5_THRESH = {FCST_VAR1_THRESH}

OBS_VAR1_NAME = PP_probs
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_THRESH = ge0.02, ge0.05, ge0.10, ge0.10, ge0.15, ge0.30, ge0.45, ge0.60

OBS_VAR2_NAME = {OBS_VAR1_NAME}
OBS_VAR2_LEVELS = {OBS_VAR1_LEVELS}
OBS_VAR2_THRESH = {OBS_VAR1_THRESH}
```

(continues on next page)

(continued from previous page)

```
OBS_VAR3_NAME = {OBS_VAR1_NAME}
OBS_VAR3_LEVELS = {OBS_VAR1_LEVELS}
OBS_VAR3_THRESH = {OBS_VAR1_THRESH}

OBS_VAR4_NAME = {OBS_VAR1_NAME}
OBS_VAR4_LEVELS = {OBS_VAR1_LEVELS}
OBS_VAR4_THRESH = {OBS_VAR1_THRESH}

OBS_VAR5_NAME = {OBS_VAR1_NAME}
OBS_VAR5_LEVELS = {OBS_VAR1_LEVELS}
OBS_VAR5_THRESH = {OBS_VAR1_THRESH}


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_OUTPUT_FLAG_CTC = BOTH
GRID_STAT_OUTPUT_FLAG_CTS = BOTH

GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
```

(continues on next page)

(continued from previous page)

```
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
```

(continues on next page)

(continued from previous page)

```
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstHRRR_obsPracPerfect_SurrogateSevere.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/
↪GridStat_fcstHRRR_obsPracPerfect_SurrogateSevere.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstHRRR_obsPracPerfect_SurrogateSevere.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/
↪GridStat_fcstHRRR_obsPracPerfect_SurrogateSevere.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/short_range/surrogate_severe_prac_perfect/grid_stat (relative to **OUTPUT_BASE**) and will contain the following files:

grid_stat_360000L_20200206_120000V_ctc.txt grid_stat_360000L_20200206_120000V_cts.txt
grid_stat_360000L_20200206_120000V.stat

Keywords

Note:

- GridStatToolUseCase
- ShortRangeAppUseCase
- NetCDFFileUseCase
- NOAAHWTOrgUseCase
- NCAROrgUseCase
- NOAAHMTOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-SS_PP_prob.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.4 METdbLoad: Brightness Temperature

model_applications/ short_range/ METdbLoad_fcstFV3_obsGoes_BrightnessTemp.conf

Scientific Objective

Load MET data into a database using the met_db_load.py script found in dtcenter/METdataio. Specifically, this use case loads distance map output from grid_stat and mode output into a database.

Datasets

Input: MET .stat files and MODE text files

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to see the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1765) section for more information.

METplus Components

This use case utilizes the METplus METdbLoad wrapper to search for files ending with .stat or .txt, substitute values into an XML load configuration file, and call met_db_load.py. It then loads data into a METviewer database for the following use cases: MODE_fcstFV3_obsGOES_BrightnessTemp, MODE_fcstFV3_obsGOES_BrightnessTempObjs, and GridStat_fcstFV3_obsGOES_BrightnessTempDmap.

METplus Workflow

The METdbload is run once and loads data for two ensemble members, one model initialization time and 2 forecast lead times, listed below.

Valid: 2019-05-21_01Z
Forecast lead: 01

Valid: 2019-05-21_02Z
Forecast lead: 02

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/
↪METdbLoad_fcstFV3_obsGoes_BrightnessTemp.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = METDbLoad


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID

VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2019052112
VALID_END = 2019052100
VALID_INCREMENT = 12H

MET_DB_LOAD_RUNTIME_FREQ = RUN_ONCE


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

MET_DB_LOAD_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/short_range/METdbLoad_fcstFV3_
↪obsGoes_BrightnessTemp/grid_stat,{INPUT_BASE}/model_applications/short_range/METdbLoad_
↪fcstFV3_obsGoes_BrightnessTemp/grid_stat_obj,{INPUT_BASE}/model_applications/short_range/
↪METdbLoad_fcstFV3_obsGoes_BrightnessTemp/mode


###
# METdbLoad Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#metdbload
###

MET_DATA_DB_DIR = {METPLUS_BASE}/../METdataio

# XML file with settings for
MET_DB_LOAD_XML_FILE = {PARM_BASE}/use_cases/met_tool_wrapper/METdbLoad/METdbLoadConfig.xml

# If true, remove temporary XML with values substituted from XML_FILE
# Set to false for debugging purposes
MET_DB_LOAD_REMOVE_TMP_XML = True

# connection info
MET_DB_LOAD_MV_HOST = localhost:3306
MET_DB_LOAD_MV_DATABASE = mv_metplus_test
MET_DB_LOAD_MV_USER = root
MET_DB_LOAD_MV_PASSWORD = mvuser

# data info
MET_DB_LOAD_MV_VERBOSE = false
MET_DB_LOAD_MV_INSERT_SIZE = 1
MET_DB_LOAD_MV_MODE_HEADER_DB_CHECK = false
MET_DB_LOAD_MV_DROP_INDEXES = false
MET_DB_LOAD_MV_APPLY_INDEXES = true
#MET_DB_LOAD_MV_GROUP = METplus Input Test
MET_DB_LOAD_MV_GROUP = Testing
MET_DB_LOAD_MV_LOAD_STAT = true
MET_DB_LOAD_MV_LOAD_MODE = true
MET_DB_LOAD_MV_LOAD_MTD = false
MET_DB_LOAD_MV_LOAD_MPR = false
```

XML Configuration

METplus substitutes values in the template XML configuration file based on user settings in the METplus configuration file. While the XML template may appear to reference environment variables, this is not actually the case. These strings are used as a reference for the wrapper to substitute values.

Note: See the *METdbLoad XML Configuration* (page 169) section of the User's Guide for more information on the values substituted in the file below:

```
<load_spec>
  <connection>
    <host>${METPLUS_MV_HOST}</host>
    <database>${METPLUS_MV_DATABASE}</database>
    <user>${METPLUS_MV_USER}</user>
    <password>${METPLUS_MV_PASSWORD}</password>
  </connection>

  <verbose>${METPLUS_MV_VERBOSE}</verbose>
  <insert_size>${METPLUS_MV_INSERT_SIZE}</insert_size>
  <mode_header_db_check>${METPLUS_MV_MODE_HEADER_DB_CHECK}</mode_header_db_check>
  <drop_indexes>${METPLUS_MV_DROP_INDEXES}</drop_indexes>
  <apply_indexes>${METPLUS_MV_APPLY_INDEXES}</apply_indexes>
  <group>${METPLUS_MV_GROUP}</group>
  <load_stat>${METPLUS_MV_LOAD_STAT}</load_stat>
  <load_mode>${METPLUS_MV_LOAD_MODE}</load_mode>
  <load_mtd>${METPLUS_MV_LOAD_MTD}</load_mtd>
  <load_mpr>${METPLUS_MV_LOAD_MPR}</load_mpr>

  <folder_tmpl>{dirs}</folder_tmpl>
  <load_val>
    <field name="dirs">
      ${METPLUS_INPUT_PATHS}
    </field>
  </load_val>
</load_spec>
```

Running METplus

This use case can be run two ways:

1) Passing in METdbLoad_fcstFV3_obsGoes_BrightnessTemp.conf followed by a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/short_range/METdbLoad_
↪fcstFV3_obsGoes_BrightnessTemp.conf /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config and then passing in METdbLoad_fcstFV3_obsGoes_BrightnessTemp.conf:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/short_range/METdbLoad_
↪fcstFV3_obsGoes_BrightnessTemp.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path to directory where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[config]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Output files are not generated. Rather, data should be available in the METviewer database. The data in the database should include Stat data for two variables and two model ensembles, and mode data.

Keywords

Note:

- METdbLoadUseCase
- ShortRangeAppUseCase
- NOAAEMCOrgUseCase
- NOAAHWTOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-METdbLoad_fcstFV3_obsGoes_BrightnessTemp.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.5 Grid-Stat: Brightness Temperature Distance Maps

model_applications/ short_range/ GridStat_fcstFV3_obsGOES_BrightnessTempDmap.conf

Scientific Objective

To provide statistical inforation on regions of low brightness temperatures, defined by creating distance maps on the FV3 ensemble members compared to GOES channel 13 brightness temperature satellite data.

Datasets

- Forecast dataset: FV3 Model member data
- Observation dataset: GOES Brightness Temperature

METplus Components

This use case runs runs grid_stat to compute distance maps using a brightness temperature less than 235 K for the forecast and observations.

METplus Workflow

The GridStat tool is run for each of 2 ensemble members and for each time. This example loops by initialization time. It processes 2 lead times, listed below.

Valid: 2019-05-21_01Z
Forecast lead: 01

Valid: 2019-05-21_02Z
Forecast lead: 02

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/short_range/GridStat_fcstFV3_obsGOES_BrightnessTempDmap.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/GridStat_
↪fcstFV3_obsGOES_BrightnessTempDmap.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat(lsm1), GridStat(mp1)


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = init
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2019052100
INIT_END = 2019052100
INIT_INCREMENT = 3600

LEAD_SEQ = 1,2


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/brightness_temperature
FCST_GRID_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/core_{instance}/core_{instance}_{init?
↪fmt=%Y%m%d}_{init?fmt=%H%M}_f{lead?fmt=%HH}.nc

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/brightness_temperature
OBS_GRID_STAT_INPUT_TEMPLATE = {valid?fmt=%Y_%m_%d}_141/remap_GOES-16.{valid?fmt=%Y%m%d}.
↪{valid?fmt=%H%M%S}.nc

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/short_range/brightness_temperature/grid_stat



###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

FCST_GRID_STAT_VAR1_NAME = SBTA1613_topofatmosphere
FCST_GRID_STAT_VAR1_LEVELS = "(*,*)"
FCST_GRID_STAT_VAR1_THRESH = le235
FCST_GRID_STAT_VAR1_OPTIONS = file_type = NETCDF_MET;

OBS_GRID_STAT_VAR1_NAME = channel_13_brightness_temperature
OBS_GRID_STAT_VAR1_LEVELS = "(*,*)"
OBS_GRID_STAT_VAR1_THRESH = le235
OBS_GRID_STAT_VAR1_OPTIONS = file_type = NETCDF_MET;

###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_OUTPUT_PREFIX = FV3_core_{instance}

GRID_STAT_OUTPUT_FLAG_DMAP = BOTH

GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_DISTANCE_MAP = TRUE
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
```

(continues on next page)

(continued from previous page)

```
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
```

(continues on next page)

(continued from previous page)

```
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstFV3_obsGOES_BrightnessTempDmap.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/
↪GridStat_fcstFV3_obsGOES_BrightnessTempDmap.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstFV3_obsGOES_BrightnessTempDmap.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/
↪GridStat_fcstFV3_obsGOES_BrightnessTempDmap.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify

the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in short_range/brightness_temperature (relative to **OUTPUT_BASE**) and will contain the following files:

grid_stat/grid_stat_FV3_core_lsm1_000000L_20190521_010000V_dmap.txt grid_stat/grid_stat_FV3_core_lsm1_0000
grid_stat/grid_stat_FV3_core_lsm1_000000L_20190521_010000V.stat grid_stat/grid_stat_FV3_core_lsm1_000000L_2
grid_stat/grid_stat_FV3_core_lsm1_000000L_20190521_020000V_pairs.nc grid_stat/grid_stat_FV3_core_lsm1_00000
grid_stat/grid_stat_FV3_core_mp1_000000L_20190521_010000V_dmap.txt grid_stat/grid_stat_FV3_core_mp1_0000
grid_stat/grid_stat_FV3_core_mp1_000000L_20190521_010000V.stat grid_stat/grid_stat_FV3_core_mp1_000000L_2
grid_stat/grid_stat_FV3_core_mp1_000000L_20190521_020000V_pairs.nc grid_stat/grid_stat_FV3_core_mp1_00000

Keywords

Note:

- GridStatToolUseCase
- ShortRangeAppUseCase
- NetCDFFileUseCase
- NOAAEMCOrgUseCase
- NOAAHWTOrgUseCase

- ValidationUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/short_range-GridStat_fcstFV3_obsGOES_BrightnessTempDmap.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.6 UserScript: Physics Tendency Vertical Cross Section plot

model_applications/short_range/UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalCrossSection.conf

Scientific Objective

To plot tendencies of temperature, moisture, and wind components averaged over a time window and spatial domain. Tendencies are partitioned into physics parameterizations and dynamics. Physics parameterizations include schemes like deep convection, convective gravity wave drag, short wave radiation, plantetary boundary layer, microphysics, and others. Non-physics tendencies (or dynamics) are due to horizontal and vertical motion. The residual (which should be zero) is the difference between the actual change in the state variable over the requested time window and the expected change due to physics parameterizations and dynamics tendencies. One can plot a single tendency component at multiple pressure levels or plot all tendency components at a single pressure level. This use case illustrates how to generate the vertical cross section plot. The METplotpy source code is needed to generate the plot. Clone the METplotpy repository (https://github.com/dtcenter/METplotpy) under the same base directory as the METPLUS_BASE directory so that the METplus and METplotpy directories are under the same base directory (i.e. if the METPLUS_BASE directory is /home/username/working/METplus, then clone the METplotpy source code into the /home/username/working directory).

Datasets

Forecast dataset: FV3 3-D history file with physics and dynamics tendencies
Grid specification: FV3 2-D grid specification file with latitude and longitude of each grid point

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1779) section for more information.

External Dependencies

You will need to use a versio of Python 3.86 that has the following packages installed:

- cartopy (0.20.3 only)
- matplotlib
- metpy
- numpy
- pandas
- shapely
- xarray

METplus Components

This use case runs the METplotpy cross_section_vert.py script to generate the plan views.

METplus Workflow

This use case does not loop but plots physics tendency data that has been subsetted to one date: 2019-05-04.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line i.e. parm/use_cases/model_applications/short_range/UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalCrossSection

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/
↪UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalCrossSection.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript
```

(continues on next page)

(continued from previous page)

```
###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = REALTIME
VALID_TIME_FMT = %Y
VALID_BEG = 2019

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

FV3_HISTORY_FILE = {INPUT_BASE}/model_applications/short_range/UserScript_fcstFV3_fcstOnly_
↪PhysicsTendency/fv3_history.nc
GRID_SPEC_FILE = {INPUT_BASE}/model_applications/short_range/UserScript_fcstFV3_fcstOnly_
↪PhysicsTendency/grid_spec.nc
USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/model_applications/short_range/UserScript_
↪fcstFV3_fcstOnly_PhysicsTendency_VerticalCrossSection/vertical_cross_section_plot.py {PARM_
↪BASE}/use_cases/model_applications/short_range/UserScript_fcstFV3_fcstOnly_PhysicsTendency_
↪VerticalCrossSection/physics_tendency_vertical_cross_section.yaml {FV3_HISTORY_FILE} {GRID_
↪SPEC_FILE} tmp -t 2 -v 20190504T14 -s 32 -115 -e 34 -82 -o {OUTPUT_BASE}/plots/short_range-
↪physics_tendency_vertical_cross_section.png --nofineprint

[user_env_vars]

# Vertical Cross Section plot specific variables

LOG_FILE = "VerticalCrossSection.log"

LOG_LEVEL = "DEBUG"
```

(continues on next page)

(continued from previous page)

```
YAML_CONFIG_NAME = {PARM_BASE}/use_cases/model_applications/short_range/UserScript_fcstFV3_
↪fcstOnly_PhysicsTendency_VerticalCrossSection/physics_tendency_vertical_cross_section.yaml
```

MET Configuration

No MET tools are used in this use case.

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

Note: See the *MODE MET Configuration* (page 174) section of the User's Guide for more information on the environment variables used in the file below:

Running METplus

This use case can be run in the following way:

1) Passing in UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalCrossSection.conf then a user-specific system configuration file:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/short_range/
↪UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalCrossSection.conf  /path/to/user_
↪system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalCrossSection.conf:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/short_range/
↪UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalCrossSection.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. The following file will be created:

short_range-physics_tendency_vertical_cross_section.png

Keywords

Note:

- MediumRangeAppUseCase
- PhysicsTendency
- ValidationUseCase
- ShortRangeAppUseCase
- S2SAppUseCase
- METplotpyUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-UserScript_fcstFV3_fcstOnly_PhysicsTendency_VerticalCrossSec

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.7 MODE: Multivariate

model_applications/ short_range/ MODEMultivar_fcstHRRR_obsMRMS_HRRRanl.conf

Scientific Objective

This use case demonstrates how to run Multivariate MODE to identify complex objects from two or more fields defined by a logical expression. This use case identifies blizzard-like objects defined by the intersection of : 1) the presence of snow precipitation type, 2) 10-m winds > 20 mph, and 3) visibility < 1/2 mile. The use of multivariate MODE is well-suited to assess the structure and placement of complex high-impact events such as blizzard conditions and heavy snow bands. Output from this use-case consists of the MODE ASCII, NetCDF, and PostScript files for the MODE forecast and observation super objects.

In this case, MODE super object intensity statistics were ouput for both 10-m wind and visibility. Using the the MODE_MULTIVAR_INTENSITY_FLAG, the user can control for which variables super object intensity statistics will be output. If all are set to False, then no intensity information will be output and only statistics relative to the super-object geometry will be available. In the case no requested intesities, the parameters MODE_FCST/OBS_MULTIVAR_NAME and/or MODE_FCST/OBS_MULTIVAR_LEVEL may be used as identifiers for the super-object.

Datasets

Forecast dataset: 1-hour HRRR in grib2

Observation dataset: MRMS and HRRR analysis in grib2

The forecast and observation fields are only a subset of the full domain in order for a faster run-time of Multivariate MODE. An example command using wgrib2 to create the HRRR subdomain is:

```
wgrib2 infile.grib2 -new_grid_winds earth -new_grid lambert:262.5:38.5:38.5:38.5 -83.
↪0:400:3000 37.0:400:3000 outfile.grib2
```

Location: All of the input data required for this use case can be found in the *short_range* sample data tarball. Navigate to METplus Releases and download sample data for the appropriate release.

This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 26) for more information.

METplus Components

This use case utilizes the METplus MODE wrapper, ingesting multiple variables to output complex super objects based on a user-defined logical expression.

METplus Workflow

MODE is the only tool called and ingests multiple fields to create a complex super object.

This example runs a single forecast hour.

Initialization: 2021020100
Forecast lead: 21

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line: parm/use_cases/model_applications/short_range/MODEMultivar_fcstHRRR_obsMRMS_HRRRanl.conf

```
[config]

# Documentation for this use-case can be found at:
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/
↪MODEMultivar_fcstHRRR_obsMRMS_HRRRanl.html

# Processes to run
PROCESS_LIST = MODE

# Time Info
LOOP_ORDER = times
LOOP_BY = INIT

INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2021020100
INIT_END = 2021020100

LEAD_SEQ = 21

MODEL = HRRR
OBTYPE = ANALYSIS

#################################
# Multivariate MODE Configurations
#################################

# Run MODE to output super objects
MODE_MULTIVAR_LOGIC = #1 && #2 && #3
MODE_MULTIVAR_INTENSITY_FLAG = FALSE,TRUE,TRUE
```

(continues on next page)

(continued from previous page)

```
FCST_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/MODEMultivar_fcstHRRR_
↪obsMRMS_HRRRanl
FCST_MODE_INPUT_TEMPLATE = hrrr.t{init?fmt=%H}z.wrfprsf{lead?fmt=%H}.sub.grib2,hrrr.t{init?
↪fmt=%H}z.wrfprsf{lead?fmt=%H}.sub.grib2,hrrr.t{init?fmt=%H}z.wrfprsf{lead?fmt=%H}.sub.grib2

OBS_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/MODEMultivar_fcstHRRR_
↪obsMRMS_HRRRanl
OBS_MODE_INPUT_TEMPLATE = PrecipFlag_00.00_{valid?fmt=%Y%m%d}-{valid?fmt=%2H}0000.sub.grib2,
↪hrrr.t{valid?fmt=%H}z.wrfprsf00.sub.grib2,hrrr.t{valid?fmt=%H}z.wrfprsf00.sub.grib2

MODE_OUTPUT_DIR = {OUTPUT_BASE}/mode
MODE_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/f{lead?fmt=%2H}

FCST_VAR1_NAME = CSNOW
FCST_VAR1_LEVELS = L0
FCST_VAR1_OPTIONS = conv_radius = 0; conv_thresh = ==1; merge_flag = NONE

OBS_VAR1_NAME = PrecipFlag
OBS_VAR1_LEVELS = L0
OBS_VAR1_OPTIONS = conv_radius = 0; conv_thresh = ==3; merge_flag = NONE

FCST_VAR2_NAME = VIS
FCST_VAR2_LEVELS = L0
FCST_VAR2_OPTIONS = conv_radius = 5; conv_thresh = <=804.672; merge_thresh = <=1207.008;␣
↪merge_flag = THRESH

OBS_VAR2_NAME = VIS
OBS_VAR2_LEVELS = L0
OBS_VAR2_OPTIONS = conv_radius = 5; conv_thresh = <=804.672; merge_thresh = <=1207.008;␣
↪merge_flag = THRESH

FCST_VAR3_NAME = WIND
FCST_VAR3_LEVELS = Z10
FCST_VAR3_OPTIONS = conv_radius = 5; conv_thresh = >=8.9408; merge_thresh = >=6.7056; merge_
↪flag = THRESH

OBS_VAR3_NAME = WIND
OBS_VAR3_LEVELS = Z10
OBS_VAR3_OPTIONS = conv_radius = 5; conv_thresh = >=8.9408; merge_thresh = >=6.7056; merge_
↪flag = THRESH

MODE_FCST_FILTER_ATTR_NAME = AREA
MODE_FCST_FILTER_ATTR_THRESH = >=25
MODE_OBS_FILTER_ATTR_NAME = AREA
MODE_OBS_FILTER_ATTR_THRESH = >=25
```

(continues on next page)

(continued from previous page)

```
MODE_MATCH_FLAG = MERGE_BOTH

MODE_REGRID_TO_GRID = FCST
MODE_REGRID_METHOD = NEAREST
MODE_REGRID_WIDTH = 1
MODE_REGRID_VLD_THRESH = 0.5

MODE_OUTPUT_PREFIX = {MODEL}_vs_{OBTYPE}

MODE_GRID_RES = 3

MODE_NC_PAIRS_FLAG_LATLON = TRUE
MODE_NC_PAIRS_FLAG_RAW = TRUE
MODE_NC_PAIRS_FLAG_OBJECT_RAW = TRUE
MODE_NC_PAIRS_FLAG_OBJECT_ID = TRUE
MODE_NC_PAIRS_FLAG_CLUSTER_ID = TRUE
MODE_NC_PAIRS_FLAG_POLYLINES = TRUE

MODE_QUILT = False

MODE_PS_PLOT_FLAG = TRUE
MODE_CT_STATS_FLAG = TRUE
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *MODE MET Configuration* (page 174) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Approximate grid resolution (km)
//
// grid_res =
${METPLUS_GRID_RES}

////////////////////////////////////////////////////////////////////////////////

//
// Run all permutations of radius and threshold
//
// quilt =
${METPLUS_QUILT}

//
```

(continues on next page)

(continued from previous page)

```
// MODE Multivar boolean combination logic
//
//multivar_logic =
${METPLUS_MULTIVAR_LOGIC}

//multivar_intensity_flag =
${METPLUS_MULTIVAR_INTENSITY_FLAG}

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FIELD}

   ${METPLUS_FCST_CENSOR_THRESH}
   ${METPLUS_FCST_CENSOR_VAL}
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}
   ${METPLUS_FCST_VLD_THRESH}
   ${METPLUS_FCST_FILTER_ATTR_NAME}
   ${METPLUS_FCST_FILTER_ATTR_THRESH}
   ${METPLUS_FCST_MERGE_THRESH}
   ${METPLUS_FCST_MERGE_FLAG}
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_MULTIVAR_NAME}
   ${METPLUS_FCST_MULTIVAR_LEVEL}
}

obs = {
   ${METPLUS_OBS_FIELD}

   ${METPLUS_OBS_CENSOR_THRESH}
   ${METPLUS_OBS_CENSOR_VAL}
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
   ${METPLUS_OBS_VLD_THRESH}
   ${METPLUS_OBS_FILTER_ATTR_NAME}
   ${METPLUS_OBS_FILTER_ATTR_THRESH}
   ${METPLUS_OBS_MERGE_THRESH}
   ${METPLUS_OBS_MERGE_FLAG}
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_MULTIVAR_NAME}
   ${METPLUS_OBS_MULTIVAR_LEVEL}
}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Handle missing data
//
// mask_missing_flag =
${METPLUS_MASK_MISSING_FLAG}

//
// Match objects between the forecast and observation fields
//
//match_flag =
${METPLUS_MATCH_FLAG}

//
// Maximum centroid distance for objects to be compared
//
//max_centroid_dist =
${METPLUS_MAX_CENTROID_DIST}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine weights
//
//weight = {
${METPLUS_WEIGHT_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine interest functions
//
interest_function = {

   ${METPLUS_INTEREST_FUNCTION_CENTROID_DIST}
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_INTEREST_FUNCTION_BOUNDARY_DIST}

   ${METPLUS_INTEREST_FUNCTION_CONVEX_HULL_DIST}

   angle_diff = (
      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 0.0 )
   );

   aspect_diff = (
      (  0.00, 1.0 )
      (  0.10, 1.0 )
      (  0.75, 0.0 )
   );

   corner   = 0.8;
   ratio_if = (
      (    0.0, 0.0 )
      ( corner, 1.0 )
      (    1.0, 1.0 )
   );

   area_ratio = ratio_if;

   int_area_ratio = (
      ( 0.00, 0.00 )
      ( 0.10, 0.50 )
      ( 0.25, 1.00 )
      ( 1.00, 1.00 )
   );

   curvature_ratio = ratio_if;

   complexity_ratio = ratio_if;

   inten_perc_ratio = ratio_if;
}

////////////////////////////////////////////////////////////////////////////////

//
// Total interest threshold for determining matches
//
//total_interest_thresh =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_TOTAL_INTEREST_THRESH}

//
// Interest threshold for printing output pair information
//
print_interest_thresh = 0.0;

////////////////////////////////////////////////////////////////////////////////

//
// Plotting information
//
met_data_dir = "MET_BASE";

fcst_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

obs_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

object_plot = {
   color_table      = "MET_BASE/colortables/mode_obj.ctable";
}

//
// Boolean for plotting on the region of valid data within the domain
//
plot_valid_flag = FALSE;

//
// Plot polyline edges using great circle arcs instead of straight lines
//
plot_gcarc_flag = FALSE;

////////////////////////////////////////////////////////////////////////////////

//
// NetCDF matched pairs, PostScript, and contingency table output files
//
```

(continues on next page)

(continued from previous page)

```
//ps_plot_flag =
${METPLUS_PS_PLOT_FLAG}

//nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

//ct_stats_flag =
${METPLUS_CT_STATS_FLAG}


////////////////////////////////////////////////////////////////////////////////

shift_right = 0;   //  grid squares

////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

Pass the use case configuration file to the run_metplus.py script along with any user-specific system configuration files if desired:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/short_range/MODEMultivar_
↪fcstHRRR_obsMRMS_HRRRanl.conf /path/to/user_system.conf
```

See *Running METplus* (page 26) for more information.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in OUTPUT_BASE and will contain the following files in the directory mode/2021020100/f21:

- mode_Fcst_VIS_L0_Obs_VIS_L0_HRRR_vs_ANALYSIS_210000L_20210201_210000V_000000A_cts.txt
- mode_Fcst_VIS_L0_Obs_VIS_L0_HRRR_vs_ANALYSIS_210000L_20210201_210000V_000000A_obj.nc
- mode_Fcst_VIS_L0_Obs_VIS_L0_HRRR_vs_ANALYSIS_210000L_20210201_210000V_000000A_obj.txt
- mode_Fcst_VIS_L0_Obs_VIS_L0_HRRR_vs_ANALYSIS_210000L_20210201_210000V_000000A.ps
- mode_Fcst_WIND_Z10_Obs_WIND_Z10_HRRR_vs_ANALYSIS_210000L_20210201_210000V_000000A_cts.txt
- mode_Fcst_WIND_Z10_Obs_WIND_Z10_HRRR_vs_ANALYSIS_210000L_20210201_210000V_000000A_obj.nc
- mode_Fcst_WIND_Z10_Obs_WIND_Z10_HRRR_vs_ANALYSIS_210000L_20210201_210000V_000000A_obj.txt
- mode_Fcst_WIND_Z10_Obs_WIND_Z10_HRRR_vs_ANALYSIS_210000L_20210201_210000V_000000A.ps

Keywords

Note:

- MODEToolUseCase
- ShortRangeAppUseCase
- GRIB2FileUseCase
- RegriddingInToolUseCase
- NOAAWPCOrgUseCase
- NCAROrgUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-MODEMultivar_fcstHRRR_obsMRMS_HRRRanl.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.8 MODE: Hail Verification

model_applications/ short_range/ MODE_fcstHRRR_obsMRMS_Hail_GRIB2.conf

Scientific Objective

To provide statistical inforation on the forecast hail size compared to the observed hail size from MRMS MESH data. Using objects to verify hail size avoids the "unfair penalty" issue, where a CAM must first generate convection to have any chance of accurately predicting the hail size. In addition, studies have shown that MRMS MESH observed hail sizes do not correlate one- to-one with observed sizes but can only be used to group storms into general categories. Running MODE allows a user to do this.

Datasets

- Forecast dataset: HRRRv4 data
- Observation dataset: MRMS

METplus Components

This use case runs MODE to create object statistics on forecast hail size from the HRRR version 4 model and the observed MRMS MESH hail size.

METplus Workflow

The MODE tool is run for each time. This example loops by valid time. It processes 2 valid times, listed below.

Valid: 2019-05-29_02Z
Forecast lead: 26

Valid: 2019-05-29_03Z
Forecast lead: 27

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/short_range/MODE_fcstHRRR_obsMRMS_Hail_GRIB2.conf

```
[config]

# Documentation for this use case can be found at
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/MODE_
↪fcstHRRR_obsMRMS_Hail_GRIB2.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = MODE


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = valid
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2019052902
VALID_END = 2019052903
VALID_INCREMENT = 3600

INIT_SEQ = 0
LEAD_SEQ_MAX = 36
LEAD_SEQ_MIN = 12


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/hrrr_esrl
FCST_MODE_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/hrrr_esrl_{init?fmt=%Y%m%d%H}f{lead?fmt=%HHH}.
```

(continues on next page)

(continued from previous page)

```
↪grib2

OBS_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/hrrr_esrl
OBS_MODE_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/mrms.MESH_Max_60min.{valid?fmt=%Y%m%d}_{valid?
↪fmt=%H%M%S}.grib2

MODE_OUTPUT_DIR = {OUTPUT_BASE}/hailtest
MODE_VERIFICATION_MASK_TEMPLATE = {FCST_MODE_INPUT_DIR}/{init?fmt=%Y%m%d}_hrefv2_
↪subdomainmask.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = HRRRv4_HAILCAST
FCST_VAR1_NAME = HAIL
FCST_VAR1_LEVELS = L0
FCST_VAR1_OPTIONS = convert(x) = x / 0.0254
MODE_FCST_CENSOR_THRESH = >0&&<0.75
MODE_FCST_CENSOR_VAL = -9999.0
MODE_FCST_FILTER_ATTR_NAME = AREA
MODE_FCST_FILTER_ATTR_THRESH = >=4

OBTYPE = MRMS
OBS_VAR1_NAME = MESHMax60min
OBS_VAR1_LEVELS = Z500
OBS_VAR1_OPTIONS = convert(x) = MM_to_IN(x);
MODE_OBS_CENSOR_THRESH = >0&&<0.75
MODE_OBS_CENSOR_VAL = -9999.0
MODE_OBS_FILTER_ATTR_NAME = AREA
MODE_OBS_FILTER_ATTR_THRESH = >=4


###
# MODE Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mode
###

MODE_QUILT = True

MODE_CONV_RADIUS = 4

MODE_CONV_THRESH = >=0.5
```

(continues on next page)

(continued from previous page)

```
MODE_MERGE_THRESH = >=0.0

MODE_MERGE_FLAG = NONE

MODE_MATCH_FLAG = NO_MERGE

MODE_MAX_CENTROID_DIST = 400.0/grid_res

MODE_MASK_MISSING_FLAG = BOTH

MODE_MASK_POLY_FLAG = BOTH

MODE_WEIGHT_INTEN_PERC_VALUE = 99

MODE_TOTAL_INTEREST_THRESH = 0.5

MODE_REGRID_TO_GRID = FCST
MODE_REGRID_METHOD = MAX
MODE_REGRID_WIDTH = 2
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *MODE MET Configuration* (page 174) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Approximate grid resolution (km)
//
// grid_res =
${METPLUS_GRID_RES}

////////////////////////////////////////////////////////////////////////////////

//
// Run all permutations of radius and threshold
//
// quilt =
${METPLUS_QUILT}

//
// MODE Multivar boolean combination logic
//
//multivar_logic =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_MULTIVAR_LOGIC}

//multivar_intensity_flag =
${METPLUS_MULTIVAR_INTENSITY_FLAG}

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FIELD}

   ${METPLUS_FCST_CENSOR_THRESH}
   ${METPLUS_FCST_CENSOR_VAL}
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}
   ${METPLUS_FCST_VLD_THRESH}
   ${METPLUS_FCST_FILTER_ATTR_NAME}
   ${METPLUS_FCST_FILTER_ATTR_THRESH}
   ${METPLUS_FCST_MERGE_THRESH}
   ${METPLUS_FCST_MERGE_FLAG}
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_MULTIVAR_NAME}
   ${METPLUS_FCST_MULTIVAR_LEVEL}
}

obs = {
   ${METPLUS_OBS_FIELD}

   ${METPLUS_OBS_CENSOR_THRESH}
   ${METPLUS_OBS_CENSOR_VAL}
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
   ${METPLUS_OBS_VLD_THRESH}
   ${METPLUS_OBS_FILTER_ATTR_NAME}
   ${METPLUS_OBS_FILTER_ATTR_THRESH}
   ${METPLUS_OBS_MERGE_THRESH}
   ${METPLUS_OBS_MERGE_FLAG}
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_MULTIVAR_NAME}
   ${METPLUS_OBS_MULTIVAR_LEVEL}
}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Handle missing data
//
// mask_missing_flag =
${METPLUS_MASK_MISSING_FLAG}

//
// Match objects between the forecast and observation fields
//
//match_flag =
${METPLUS_MATCH_FLAG}

//
// Maximum centroid distance for objects to be compared
//
//max_centroid_dist =
${METPLUS_MAX_CENTROID_DIST}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine weights
//
//weight = {
${METPLUS_WEIGHT_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine interest functions
//
interest_function = {

   ${METPLUS_INTEREST_FUNCTION_CENTROID_DIST}

   ${METPLUS_INTEREST_FUNCTION_BOUNDARY_DIST}

   ${METPLUS_INTEREST_FUNCTION_CONVEX_HULL_DIST}
```

(continues on next page)

(continued from previous page)

```
   angle_diff = (
      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 0.0 )
   );

   aspect_diff = (
      (  0.00, 1.0 )
      (  0.10, 1.0 )
      (  0.75, 0.0 )
   );

   corner   = 0.8;
   ratio_if = (
      (    0.0, 0.0 )
      ( corner, 1.0 )
      (    1.0, 1.0 )
   );

   area_ratio = ratio_if;

   int_area_ratio = (
      ( 0.00, 0.00 )
      ( 0.10, 0.50 )
      ( 0.25, 1.00 )
      ( 1.00, 1.00 )
   );

   curvature_ratio = ratio_if;

   complexity_ratio = ratio_if;

   inten_perc_ratio = ratio_if;
}

////////////////////////////////////////////////////////////////////////////////

//
// Total interest threshold for determining matches
//
//total_interest_thresh =
${METPLUS_TOTAL_INTEREST_THRESH}

//
```

(continues on next page)

(continued from previous page)

```
// Interest threshold for printing output pair information
//
print_interest_thresh = 0.0;

////////////////////////////////////////////////////////////////////////////////

//
// Plotting information
//
met_data_dir = "MET_BASE";

fcst_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

obs_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

object_plot = {
   color_table      = "MET_BASE/colortables/mode_obj.ctable";
}

//
// Boolean for plotting on the region of valid data within the domain
//
plot_valid_flag = FALSE;

//
// Plot polyline edges using great circle arcs instead of straight lines
//
plot_gcarc_flag = FALSE;

////////////////////////////////////////////////////////////////////////////////

//
// NetCDF matched pairs, PostScript, and contingency table output files
//
//ps_plot_flag =
${METPLUS_PS_PLOT_FLAG}
```

(continues on next page)

(continued from previous page)

```
//nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

//ct_stats_flag =
${METPLUS_CT_STATS_FLAG}


////////////////////////////////////////////////////////////////////////////////

shift_right = 0;   //  grid squares

////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in MODE_fcstHRRRE_obsMRMS_Hail_GRIB2.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/MODE_
↪fcstHRRRE_obsMRMS_Hail_GRIB2.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in MODE_fcstHRRRE_obsMRMS_Hail_GRIB2.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/MODE_
↪fcstHRRRE_obsMRMS_Hail_GRIB2.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in hailtest (relative to **OUTPUT_BASE**) and will contain the following files:

mode_260000L_20190529_020000V_010000A_cts.txt mode_260000L_20190529_020000V_010000A_obj.nc
mode_260000L_20190529_020000V_010000A_obj.txt mode_260000L_20190529_020000V_010000A.ps
mode_270000L_20190529_030000V_010000A_cts.txt mode_270000L_20190529_030000V_010000A_obj.nc
mode_270000L_20190529_030000V_010000A_obj.txt mode_270000L_20190529_030000V_010000A.ps

Keywords

Note:

- MODEToolUseCase
- ShortRangeAppUseCase
- GRIB2FileUseCase
- RegriddingInToolUseCase
- NOAAHWTOrgUseCase
- NCAROrgUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-MODE_fcstHRRRE_obsMRMS_Hail_GRIB2.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.9 Point2Grid: Calculate Practically Perfect Probabilities

model_applications/ short_range/ Point2Grid_obsLSR_ObsOnly_PracticallyPerfect.conf

Scientific Objective

To use storm reports as observations to calculate Practically Perfect probabilities.

Datasets

Relevant information about the datasets that would be beneficial include:

- Observation dataset: Local Storm Reports

METplus Components

This use case runs ASCII2NC to get the storm reports in netcdf format, runs Point2Grid to get those netcdf observations onto a grid, runs RegridDataPlane to use that gridded data as a mask to calculate probabilities

METplus Workflow

The following tools are used for each run time:

ASCII2NC > Point2Grid > RegridDataPlane

This example runs on a single time/file at a time. Each storm report is assumed to have no more than 24 hours of data inside

Run times:

2020-02-05

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/short_range/Point2Grid_obsLSR_ObsOnly_PracticallyPerfect.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/
```

(continues on next page)

(continued from previous page)

```
↪Point2Grid_obsLSR_ObsOnly_PracticallyPerfect.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = ASCII2NC, Point2Grid, RegridDataPlane


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2020020500
INIT_END = 2020020500
INIT_INCREMENT = 24H

LEAD_SEQ = 12H


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/short_range/Point2Grid_obsLSR_ObsOnly_
↪PracticallyPerfect


# ASCII2NC
```

(continues on next page)

(continued from previous page)

```
ASCII2NC_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/practically_perfect
ASCII2NC_INPUT_TEMPLATE = "{CONFIG_DIR}/read_ascii_storm.py {ASCII2NC_INPUT_DIR}/200205_rpts_
↪filtered.csv"

ASCII2NC_OUTPUT_TEMPLATE = {OUTPUT_BASE}/model_applications/short_range/practically_perfect/
↪StormReps.{init?fmt=%Y%m%d%H}.nc


# Point2Grid

POINT2GRID_INPUT_DIR = {OUTPUT_BASE}/model_applications/short_range/practically_perfect
POINT2GRID_INPUT_TEMPLATE = {OUTPUT_BASE}/model_applications/short_range/practically_perfect/
↪StormReps.{init?fmt=%Y%m%d%H}.nc

POINT2GRID_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/short_range/practically_perfect


# RegridDataPlane

OBS_REGRID_DATA_PLANE_INPUT_DIR = {POINT2GRID_OUTPUT_DIR}
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = StormReps_211.{init?fmt=%Y%m%d%H}.nc

OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {POINT2GRID_OUTPUT_DIR}
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = StormReps_211_Probs.{init?fmt=%Y%m%d}.nc

POINT2GRID_OUTPUT_TEMPLATE = {OUTPUT_BASE}/model_applications/short_range/practically_
↪perfect/StormReps_211.{init?fmt=%Y%m%d%H}.nc


###
# ASCII2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ascii2nc
###

ASCII2NC_WINDOW_BEGIN = 0
ASCII2NC_WINDOW_END = 0

ASCII2NC_INPUT_FORMAT = python
ASCII2NC_TIME_SUMMARY_FLAG = False
ASCII2NC_TIME_SUMMARY_RAW_DATA = False
ASCII2NC_TIME_SUMMARY_BEG = 000000
ASCII2NC_TIME_SUMMARY_END = 235959
ASCII2NC_TIME_SUMMARY_STEP = 300
ASCII2NC_TIME_SUMMARY_WIDTH = 600
```

(continues on next page)

(continued from previous page)

```
ASCII2NC_TIME_SUMMARY_GRIB_CODES = 11, 204, 211
ASCII2NC_TIME_SUMMARY_VAR_NAMES =
ASCII2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80
ASCII2NC_TIME_SUMMARY_VALID_FREQ = 0
ASCII2NC_TIME_SUMMARY_VALID_THRESH = 0.0


###
# Point2Grid Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#point2grid
###

POINT2GRID_REGRID_TO_GRID = G211

POINT2GRID_INPUT_FIELD =Fscale
POINT2GRID_INPUT_LEVEL =

POINT2GRID_ADP =

POINT2GRID_REGRID_METHOD = MAX

POINT2GRID_GAUSSIAN_DX = 81.271
POINT2GRID_GAUSSIAN_RADIUS = 120

POINT2GRID_PROB_CAT_THRESH =

POINT2GRID_VLD_THRESH =


###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

OBS_REGRID_DATA_PLANE_RUN = True

REGRID_DATA_PLANE_ONCE_PER_FIELD = False

OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Fscale_mask
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = "(*,*)"
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = PP_probs

REGRID_DATA_PLANE_VERIF_GRID = G211

REGRID_DATA_PLANE_METHOD = MAXGAUSS
```

(continues on next page)

(continued from previous page)

```
REGRID_DATA_PLANE_WIDTH = 1

REGRID_DATA_PLANE_GAUSSIAN_DX = 81.271
REGRID_DATA_PLANE_GAUSSIAN_RADIUS = 120
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

```
////////////////////////////////////////////////////////////////////////////////
//
// Default ascii2nc configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to summarize the ASCII data read in
//

//
// Time periods for the summarization
// obs_var (string array) is added and works like grib_code (int array)
// when the obs name is given instead of grib_code
//
${METPLUS_TIME_SUMMARY_DICT}

//
// Mapping of input little_r report types to output message types
//
message_type_map = [
   { key = "FM-12 SYNOP";  val = "ADPSFC"; },
   { key = "FM-13 SHIP";   val = "SFCSHP"; },
   { key = "FM-15 METAR";  val = "ADPSFC"; },
   { key = "FM-18 BUOY";   val = "SFCSHP"; },
   { key = "FM-281 QSCAT"; val = "ASCATW"; },
   { key = "FM-32 PILOT";  val = "ADPUPA"; },
   { key = "FM-35 TEMP";   val = "ADPUPA"; },
```

(continues on next page)

(continued from previous page)

```
   { key = "FM-88 SATOB";  val = "SATWND"; },
   { key = "FM-97 ACARS";  val = "AIRCFT"; }
];

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

See the following files for more information about the environment variables set in this configuration file.

parm/use_cases/met_tool_wrapper/Point2Grid/Point2Grid.py parm/use_cases/met_tool_wrapper/RegridDataPlane/R

Python Embedding

This use case uses a Python embedding script to read input data

parm/use_cases/model_applications/short_range/Point2Grid_obsLSR_ObsOnly_PracticallyPerfect/read_ascii_storm.py

```
import pandas as pd
import os
import sys

print(f'Python Script: {sys.argv[0]}')

# input file specified on the command line
# load the data into the numpy array

if len(sys.argv) < 2:
    script_name = os.path.basename(sys.argv[0])
    print(f"ERROR: {script_name} -> Must specify exactly one input file.")
    sys.exit(1)

# Read the input file as the first argument
input_file = os.path.expandvars(sys.argv[1])
print(f'Input File: {input_file}')

if not os.path.exists(input_file):
    print("ERROR: Could not find input file")
    sys.exit(2)
```

(continues on next page)

(continued from previous page)

```
# Read and format the input 11-column observations
COLUMN_NAMES = (
    "Message_Type",     # (1)  string
    "Station_ID",       # (2)  string
    "Valid_Time",       # (3)  string (YYYYMMDD_HHMMSS)
    "Lat",              # (4)  numeric (Deg North)
    "Lon",              # (5)  numeric (Deg East)
    "Elevation",        # (6)  numeric (msl)
    "Var_Name",         # (7)  string (or GRIB_Code)
    "Level",            # (8)  numeric
    "Height",           # (9)  numeric (msl or agl)
    "QC_String",        # (10) string
    "Observation_Value" # (11) numeric
)

# Create a blank dataframe based on the 11 column standard
point_frame = pd.DataFrame(columns=COLUMN_NAMES,dtype='str')

#Read in the Storm report, 8 columns not matching the 11 column standard
temp_data = pd.read_csv(input_file,names=['Time', 'Fscale', 'Location', 'County','Stat','Lat
↪', 'Lon', 'Comment'], dtype=str ,skiprows=1)

#Strip out any rows in the middle that are actually header rows
#Allows for concatenating storm reports together
temp_data = temp_data[temp_data["Time"] != "Time"]

#Change some columns to floats and ints
temp_data[["Lat","Lon"]] = temp_data[["Lat","Lon"]].apply(pd.to_numeric)

#Assign approprite columns to point_frame leaving missing as empty strings
point_frame["Lat"] = temp_data["Lat"]
point_frame["Lon"] = temp_data["Lon"]
#point_frame["Station_ID"] = temp_data["County"]
point_frame["Station_ID"] = "NA"
point_frame["Var_Name"] = "Fscale"
point_frame["Message_Type"] = "StormReport"

#Assign 0.0 values to numeric point_frame columns that we don't have in the csv file
point_frame["Elevation"] = 0.0
point_frame["Level"] = 0.0
point_frame["Height"] = 0.0

#Change Comments into a "QC" string Tornado=1, Hail=2, Wind=3, Other=4
point_frame["QC_String"] = "4"
```

(continues on next page)

(continued from previous page)

```
mask = temp_data["Comment"].str.contains('TORNADO')
point_frame.loc[mask,"QC_String"] = "1"
mask = temp_data["Comment"].str.contains('HAIL')
point_frame.loc[mask,"QC_String"] = "2"
mask = temp_data["Comment"].str.contains('WIND')
point_frame.loc[mask,"QC_String"] = "3"

#Time is HHMM in the csv file so we need to use a piece of the filename and
#this value to create a valid date string
file_without_path = os.path.basename(input_file)
year_month_day = "20"+file_without_path[0:6]
point_frame["Valid_Time"] = year_month_day+"_"+temp_data["Time"]+"00"

#Currently we are only interested in the fact that we have a report at that locaton
#and not its actual value so all values are 1.0
point_frame["Observation_Value"] = 1.0

#Ascii2nc wants the final values in a list
point_data = point_frame.values.tolist()

print("Data Length:\t" + repr(len(point_data)))
print("Data Type:\t" + repr(type(point_data)))

################################################################################
```

Running METplus

This use case can be run two ways:

1) Passing in EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/
↪Point2Grid_obsLSR_ObsOnly_PracticallyPerfect.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/
↪Point2Grid_obsLSR_ObsOnly_PracticallyPerfect.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in

parm/use_cases

- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/short_range/practically_perfect/ (relative to **OUTPUT_BASE**) and will contain the following files:

- StormReps_211_Probs.20200205.nc

Keywords

Note:

- ASCII2NCToolUseCase
- Point2GridToolUseCase
- RegridDataPlaneToolUseCase
- RegriddingInToolUseCase
- NetCDFFileUseCase
- PythonEmbeddingFileUseCase
- ShortRangeAppUseCase
- NCAROrgUseCase
- ProbabilityGenerationUseCase
- MaskingFeatureUseCase
- HMTOrgUseCase

- HWTOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-Point2Grid_obsLSR_ObsOnly_PracticallyPerfect.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.10 Ensemble-Stat: Ensemble Statistics using Obs Uncertainty

model_applications/ short_range/ EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField.conf

Scientific Objective

To provide useful statistical information about the ensemble characteristics such as how dispersive it is and the relationship between spread and skill. This example also shows how to compute simple probability fields called ensemble relative frequency.

Datasets

Relevant information about the datasets that would be beneficial include:

- Forecast dataset: HRRRE data
- Observation dataset: HRRRE data

METplus Components

This use case runs PB2NC on the prepBUFR observation data to convert it into NetCDF format so it can be read by MET. Then EnsembleStat is run.

METplus Workflow

The following tools are used for each run time:

PB2NC > EnsembleStat

This example loops by initialization time. For each initialization time it will process forecast leads 0, 1, and 2. There is only one initialization time in this example, so the following will be run:

Run times:

Init: 2018-07-09_12Z
Forecast lead: 0

Init: 2018-07-09_12Z
Forecast lead: 1

Init: 2018-07-09_12Z
Forecast lead: 2

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/short_range/EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/
↪EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PB2NC, EnsembleStat, GenEnsProd


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
```

(continues on next page)

(continued from previous page)

```
INIT_BEG=2018070912
INIT_END=2018070912
INIT_INCREMENT=3600

LEAD_SEQ = 0,1,2


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

# PB2NC

PB2NC_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/hrrr_ensemble_sfc/prepbufr
PB2NC_INPUT_TEMPLATE = {da_init?fmt=%Y%m%d}/{da_init?fmt=%Y%j%H%M}.rap.t{da_init?fmt=%H}z.
↪prepbufr.tm{offset?fmt=%2H}.{da_init?fmt=%Y%m%d}

PB2NC_SKIP_IF_OUTPUT_EXISTS = True

PB2NC_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/short_range/EnsembleStat_fcstHRRRE_
↪obsHRRRE_Sfc_MultiField/rap
PB2NC_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/{valid?fmt=%Y%m%d%H}.rap.nc


# EnsembleStat

FCST_ENSEMBLE_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/hrrr_ensemble_sfc/
↪fcst
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE =
    {init?fmt=%Y%m%d%H}/postprd_mem0001/wrfprs_conus_mem0001_{lead?fmt=%HH}.grib2,
    {init?fmt=%Y%m%d%H}/postprd_mem0002/wrfprs_conus_mem0002_{lead?fmt=%HH}.grib2

OBS_ENSEMBLE_STAT_POINT_INPUT_DIR = {PB2NC_OUTPUT_DIR}
OBS_ENSEMBLE_STAT_POINT_INPUT_TEMPLATE = {PB2NC_OUTPUT_TEMPLATE}

ENSEMBLE_STAT_VERIFICATION_MASK_TEMPLATE =
    {INPUT_BASE}/model_applications/short_range/mask/EAST.nc,
    {INPUT_BASE}/model_applications/short_range/mask/WEST.nc,
    {INPUT_BASE}/model_applications/short_range/mask/CONUS.nc,
    {INPUT_BASE}/model_applications/short_range/mask/LMV.nc

ENSEMBLE_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/short_range/EnsembleStat_
↪fcstHRRRE_obsHRRRE_Sfc_MultiField/EnsembleStat
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H%M}

# GenEnsProd
GEN_ENS_PROD_INPUT_DIR = {FCST_ENSEMBLE_STAT_INPUT_DIR}
GEN_ENS_PROD_INPUT_TEMPLATE = {FCST_ENSEMBLE_STAT_INPUT_TEMPLATE}

GEN_ENS_PROD_OUTPUT_DIR = {ENSEMBLE_STAT_OUTPUT_DIR}
GEN_ENS_PROD_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H%M}/gen_ens_prod_{ENSEMBLE_STAT_OUTPUT_
↪PREFIX}_{valid?fmt=%Y%m%d_%H%M%S}V_ens.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = HRRRE_ens
OBTYPE = ANALYS

ENS_VAR1_NAME = TMP
ENS_VAR1_LEVELS = Z02
ENS_VAR1_THRESH = >=283, >=288, >=293, >=298, >=303

ENS_VAR2_NAME = DPT
ENS_VAR2_LEVELS = Z2
ENS_VAR2_THRESH = >=278, >=283, >=288, >=293, >=298

ENS_VAR3_NAME = UGRD
ENS_VAR3_LEVELS = Z10
ENS_VAR3_THRESH = <=-10, <=-5, <=-2, >=2, >=5, >=10

ENS_VAR4_NAME = VGRD
ENS_VAR4_LEVELS = Z10
ENS_VAR4_THRESH = <=-10, <=-5, <=-2, >=2, >=5, >=10

ENS_VAR5_NAME = WIND
ENS_VAR5_LEVELS = Z10
ENS_VAR5_THRESH = >=2, >=4, >=6, >=8, >=10

FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = Z2
BOTH_VAR1_THRESH = >=283, >=288, >=293, >=298, >=303

OBS_VAR1_NAME = {FCST_VAR1_NAME}
OBS_VAR1_LEVELS = {FCST_VAR1_LEVELS}
```

(continues on next page)

(continued from previous page)

```
OBS_VAR1_OPTIONS = ens_ssvar_bin_size = 1.0; ens_phist_bin_size = 0.05; wind_thresh = >2.572;


###
# PB2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pb2nc
###

PB2NC_LEVEL_RANGE_END = 255

PB2NC_QUALITY_MARK_THRESH = 3

PB2NC_GRID =
PB2NC_POLY =
PB2NC_STATION_ID =
PB2NC_MESSAGE_TYPE = ADPUPA, ADPSFC, AIRCFT, PROFLR

PB2NC_OBS_BUFR_VAR_LIST = POB, QOB, TOB, ZOB, UOB, VOB, D_DPT, D_WDIR, D_WIND, D_RH, D_MIXR,␣
↪D_PRMSL

PB2NC_TIME_SUMMARY_FLAG = False
PB2NC_TIME_SUMMARY_RAW_DATA = False
PB2NC_TIME_SUMMARY_BEG = 000000
PB2NC_TIME_SUMMARY_END = 235959
PB2NC_TIME_SUMMARY_STEP = 300
PB2NC_TIME_SUMMARY_WIDTH = 600
PB2NC_TIME_SUMMARY_GRIB_CODES =
PB2NC_TIME_SUMMARY_VAR_NAMES = TMP, WDIR, RH
PB2NC_TIME_SUMMARY_TYPES =
PB2NC_TIME_SUMMARY_VALID_FREQ = 0
PB2NC_TIME_SUMMARY_VALID_THRESH = 0.0

PB2NC_OBS_WINDOW_BEGIN = -900
PB2NC_OBS_WINDOW_END = 900


###
# EnsembleStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ensemblestat
###

ENSEMBLE_STAT_OBS_WINDOW_BEGIN = -900
ENSEMBLE_STAT_OBS_WINDOW_END = 900

ENSEMBLE_STAT_N_MEMBERS = 2
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_ENS_THRESH = 1.0

ENSEMBLE_STAT_REGRID_TO_GRID = FCST
ENSEMBLE_STAT_REGRID_METHOD = BILIN
ENSEMBLE_STAT_REGRID_WIDTH = 2

ENSEMBLE_STAT_DUPLICATE_FLAG = UNIQUE
ENSEMBLE_STAT_SKIP_CONST = True

ENSEMBLE_STAT_OBS_ERROR_FLAG = TRUE

ENSEMBLE_STAT_MASK_GRID =

ENSEMBLE_STAT_CI_ALPHA = 0.01

ENSEMBLE_STAT_MESSAGE_TYPE = ADPSFC

ENSEMBLE_STAT_INTERP_METHOD = BILIN
ENSEMBLE_STAT_INTERP_WIDTH = 2

ENSEMBLE_STAT_OUTPUT_FLAG_ECNT = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_RPS = NONE
ENSEMBLE_STAT_OUTPUT_FLAG_RHIST = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_PHIST = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_ORANK = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_SSVAR = BOTH
ENSEMBLE_STAT_OUTPUT_FLAG_RELP = BOTH


ENSEMBLE_STAT_NC_ORANK_FLAG_LATLON = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_MEAN = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_RANK = FALSE
ENSEMBLE_STAT_NC_ORANK_FLAG_PIT = FALSE
ENSEMBLE_STAT_NC_ORANK_FLAG_VLD_COUNT = TRUE
ENSEMBLE_STAT_NC_ORANK_FLAG_WEIGHT = FALSE

CONFIG_DIR={PARM_BASE}/use_cases/model_applications/short_range/EnsembleStat_fcstHRRRE_
↪obsHRRRE_Sfc_MultiField
ENSEMBLE_STAT_MET_OBS_ERR_TABLE = {CONFIG_DIR}/obs_error_table_V8.0.txt

ENSEMBLE_STAT_OUTPUT_PREFIX = HRRRE_F{lead?fmt=%3H}_ADPSFC

###
# GenEnsProd Settings
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#genensprod
###


GEN_ENS_PROD_OBS_WINDOW_BEGIN = -900
GEN_ENS_PROD_OBS_WINDOW_END = 900

GEN_ENS_PROD_N_MEMBERS = 2

GEN_ENS_PROD_ENS_THRESH = 1.0

GEN_ENS_PROD_REGRID_TO_GRID = FCST
GEN_ENS_PROD_REGRID_METHOD = BILIN
GEN_ENS_PROD_REGRID_WIDTH = 2

GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_MINUS = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_PLUS = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_MIN = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_MAX = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_RANGE = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_VLD_COUNT = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_NEP = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_NMEP = FALSE
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *EnsembleStat MET Configuration* (page 105) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
```

(continues on next page)

(continued from previous page)

```
// Ensemble-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
${METPLUS_DESC}

//
// Output observation type to be written
//
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
${METPLUS_CENSOR_THRESH}
${METPLUS_CENSOR_VAL}
cat_thresh    = [];
nc_var_str    = "";

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//prob_cat_thresh =
${METPLUS_PROB_CAT_THRESH}

//prob_pct_thresh =
${METPLUS_PROB_PCT_THRESH}

//eclv_points =
${METPLUS_ECLV_POINTS}


////////////////////////////////////////////////////////////////////////////////

//
// Forecast and observation fields to be verified
//

fcst = {

   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_ENS_THRESH}
   ${METPLUS_VLD_THRESH}
   ${METPLUS_FCST_FIELD}
}

obs = {

   ${METPLUS_OBS_FILE_TYPE}

   ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//

${METPLUS_MESSAGE_TYPE}
sid_exc        = [];
//obs_thresh     = [ NA ];
${METPLUS_OBS_THRESH}
```

(continues on next page)

(continued from previous page)

```
//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

${METPLUS_DUPLICATE_FLAG}
obs_summary    = NONE;
obs_perc_value = 50;
${METPLUS_SKIP_CONST}

//
// Observation error options
// Set dist_type to NONE to use the observation error table instead
// May be set separately in each "obs.field" entry
//
obs_error = {
   ${METPLUS_OBS_ERROR_FLAG}
   dist_type        = NONE;
   dist_parm        = [];
   inst_bias_scale  = 1.0;
   inst_bias_offset = 0.0;
   min              = NA;      // Valid range of data
   max              = NA;
}

//
// Mapping of message type group name to comma-separated list of values.
//
message_type_group_map = [
   { key = "SURFACE"; val = "ADPSFC,SFCSHP,MSONET";               },
   { key = "ANYAIR";  val = "AIRCAR,AIRCFT";                      },
   { key = "ANYSFC";  val = "ADPSFC,SFCSHP,ADPUPA,PROFLR,MSONET"; },
   { key = "ONLYSF";  val = "ADPSFC,SFCSHP";                      }
];

//
// Ensemble bin sizes
// May be set separately in each "obs.field" entry
//
${METPLUS_ENS_SSVAR_BIN_SIZE}
${METPLUS_ENS_PHIST_BIN_SIZE}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}



//
// May be set separately in each "obs.field" entry
//
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
mask = {
   ${METPLUS_MASK_GRID}
   ${METPLUS_MASK_POLY}
   sid    = [];
   llpnt = [];
}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
${METPLUS_CI_ALPHA}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Interpolation methods
//
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gridded verification output types
// May be set separately in each "obs.field" entry
//
${METPLUS_NC_ORANK_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Random number generator
//
rng = {
   type = "mt19937";
   seed = "1";
}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

${METPLUS_OUTPUT_PREFIX}
//version          = "V9.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/
↪EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/
↪EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/short_range/EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField/EnsembleStat/20180709120000 (relative to **OUTPUT_BASE**) and will contain the following files:

- ensemble_stat_HRRRE_F000_ADPSFC_20180709_120000V_ecnt.txt
- ensemble_stat_HRRRE_F000_ADPSFC_20180709_120000V_ens.nc

- ensemble_stat_HRRRE_F000_ADPSFC_20180709_120000V_orank.txt
- ensemble_stat_HRRRE_F000_ADPSFC_20180709_120000V_phist.txt
- ensemble_stat_HRRRE_F000_ADPSFC_20180709_120000V_relp.txt
- ensemble_stat_HRRRE_F000_ADPSFC_20180709_120000V_rhist.txt
- ensemble_stat_HRRRE_F000_ADPSFC_20180709_120000V_ssvar.txt
- ensemble_stat_HRRRE_F000_ADPSFC_20180709_120000V.stat
- ensemble_stat_HRRRE_F001_ADPSFC_20180709_130000V_ecnt.txt
- ensemble_stat_HRRRE_F001_ADPSFC_20180709_130000V_ens.nc
- ensemble_stat_HRRRE_F001_ADPSFC_20180709_130000V_orank.txt
- ensemble_stat_HRRRE_F001_ADPSFC_20180709_130000V_phist.txt
- ensemble_stat_HRRRE_F001_ADPSFC_20180709_130000V_relp.txt
- ensemble_stat_HRRRE_F001_ADPSFC_20180709_130000V_rhist.txt
- ensemble_stat_HRRRE_F001_ADPSFC_20180709_130000V_ssvar.txt
- ensemble_stat_HRRRE_F001_ADPSFC_20180709_130000V.stat
- ensemble_stat_HRRRE_F002_ADPSFC_20180709_140000V_ecnt.txt
- ensemble_stat_HRRRE_F002_ADPSFC_20180709_140000V_ens.nc
- ensemble_stat_HRRRE_F002_ADPSFC_20180709_140000V_orank.txt
- ensemble_stat_HRRRE_F002_ADPSFC_20180709_140000V_phist.txt
- ensemble_stat_HRRRE_F002_ADPSFC_20180709_140000V_relp.txt
- ensemble_stat_HRRRE_F002_ADPSFC_20180709_140000V_rhist.txt
- ensemble_stat_HRRRE_F002_ADPSFC_20180709_140000V_ssvar.txt
- ensemble_stat_HRRRE_F002_ADPSFC_20180709_140000V.stat

Keywords

Note:

- EnsembleStatToolUseCase
- ShortRangeAppUseCase
- PB2NCToolUseCase
- prepBUFRFileUseCase
- GRIB2FileUseCase
- NCAROrgUseCase

- EnsembleAppUseCase
- ProbabilityGenerationUseCase
- NOAAGSLOrgUseCase
- DTCOrgUseCase
- ObsUncertaintyUseCase
- MaskingFeatureUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.11 MODE: Brightness Temperature Verification

model_applications/ short_range/ MODE_fcstFV3_obsGOES_BrightnessTemp.conf

Scientific Objective

To provide statistical inforation on regions of low brightness temperatures, defined by creating objects, in the FV3 model compared to GOES satellite.

Datasets

- Forecast dataset: FV3 Model member data
- Observation dataset: GOES Brightness Temperature

METplus Components

This use case runs MODE to create object statistics on brightness temperatures below 235 K.

METplus Workflow

The MODE tool is run for each of 2 ensemble members and for eachtime. This example loops by initialization time. It processes 2 lead times, listed below.

Valid: 2019-05-21_01Z
Forecast lead: 01

Valid: 2019-05-21_02Z
Forecast lead: 02

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/short_range/MODE_fcstFV3_obsGOES_BrightnessTemp.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/MODE_
↪fcstFV3_obsGOES_BrightnessTemp.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = MODE(lsm1), MODE(mp1)


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = init
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2019052100
INIT_END = 2019052100
INIT_INCREMENT = 3600
```

(continues on next page)

(continued from previous page)

```
LEAD_SEQ = 1,2


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/brightness_temperature
FCST_MODE_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/core_{instance}/core_{instance}_{init?fmt=%Y%m
↪%d}_{init?fmt=%H%M}_f{lead?fmt=%HH}.nc

OBS_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/brightness_temperature
OBS_MODE_INPUT_TEMPLATE = {valid?fmt=%Y_%m_%d}_141/remap_GOES-16.{valid?fmt=%Y%m%d}.{valid?
↪fmt=%H%M%S}.nc

MODE_OUTPUT_DIR = {OUTPUT_BASE}/short_range/brightness_temperature


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = FV3_core

FCST_VAR1_NAME = SBTA1613_topofatmosphere
FCST_VAR1_LEVELS = "(*,*)"
FCST_VAR1_OPTIONS = file_type = NETCDF_MET;

MODE_FCST_CENSOR_THRESH = <=0
MODE_FCST_CENSOR_VAL = 9999


OBTYPE = GOES

OBS_VAR1_NAME = channel_13_brightness_temperature
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_OPTIONS = file_type = NETCDF_MET;

MODE_OBS_CENSOR_THRESH = <=0
MODE_OBS_CENSOR_VAL = 9999
```

(continues on next page)

(continued from previous page)

```
###
# MODE Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mode
###

MODE_QUILT = FALSE

MODE_CONV_RADIUS = 5

MODE_CONV_THRESH = <=235

MODE_MERGE_THRESH = <=235

MODE_MERGE_FLAG = NONE

MODE_GRID_RES = 3

MODE_MAX_CENTROID_DIST = 600.0/grid_res

MODE_INTEREST_FUNCTION_CENTROID_DIST = ( ( 0.0, 1.0 ) (  60.0/grid_res, 1.0 ) ( 450.0/grid_
↪res, 0.0 ) )

MODE_MASK_MISSING_FLAG = BOTH

MODE_WEIGHT_CENTROID_DIST = 4.0
MODE_WEIGHT_BOUNDARY_DIST = 3.0
MODE_WEIGHT_CONVEX_HULL_DIST = 1.0
MODE_WEIGHT_AREA_RATIO = 4.0
MODE_WEIGHT_INT_AREA_RATIO = 3.0

MODE_TOTAL_INTEREST_THRESH = 0.65

MODE_REGRID_TO_GRID = NONE

MODE_OUTPUT_PREFIX = FV3_core_{instance}
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *MODE MET Configuration* (page 174) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Approximate grid resolution (km)
//
// grid_res =
${METPLUS_GRID_RES}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Run all permutations of radius and threshold
//
// quilt =
${METPLUS_QUILT}

//
// MODE Multivar boolean combination logic
//
//multivar_logic =
${METPLUS_MULTIVAR_LOGIC}

//multivar_intensity_flag =
${METPLUS_MULTIVAR_INTENSITY_FLAG}

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FIELD}

   ${METPLUS_FCST_CENSOR_THRESH}
   ${METPLUS_FCST_CENSOR_VAL}
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}
   ${METPLUS_FCST_VLD_THRESH}
   ${METPLUS_FCST_FILTER_ATTR_NAME}
   ${METPLUS_FCST_FILTER_ATTR_THRESH}
   ${METPLUS_FCST_MERGE_THRESH}
   ${METPLUS_FCST_MERGE_FLAG}
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_MULTIVAR_NAME}
   ${METPLUS_FCST_MULTIVAR_LEVEL}
}

obs = {
   ${METPLUS_OBS_FIELD}

   ${METPLUS_OBS_CENSOR_THRESH}
   ${METPLUS_OBS_CENSOR_VAL}
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
   ${METPLUS_OBS_VLD_THRESH}
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_OBS_FILTER_ATTR_NAME}
   ${METPLUS_OBS_FILTER_ATTR_THRESH}
   ${METPLUS_OBS_MERGE_THRESH}
   ${METPLUS_OBS_MERGE_FLAG}
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_MULTIVAR_NAME}
   ${METPLUS_OBS_MULTIVAR_LEVEL}
}

////////////////////////////////////////////////////////////////////////////////

//
// Handle missing data
//
// mask_missing_flag =
${METPLUS_MASK_MISSING_FLAG}

//
// Match objects between the forecast and observation fields
//
//match_flag =
${METPLUS_MATCH_FLAG}

//
// Maximum centroid distance for objects to be compared
//
//max_centroid_dist =
${METPLUS_MAX_CENTROID_DIST}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine weights
//
//weight = {
${METPLUS_WEIGHT_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine interest functions
//
interest_function = {

   ${METPLUS_INTEREST_FUNCTION_CENTROID_DIST}

   ${METPLUS_INTEREST_FUNCTION_BOUNDARY_DIST}

   ${METPLUS_INTEREST_FUNCTION_CONVEX_HULL_DIST}

   angle_diff = (
      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 0.0 )
   );

   aspect_diff = (
      (  0.00, 1.0 )
      (  0.10, 1.0 )
      (  0.75, 0.0 )
   );

   corner   = 0.8;
   ratio_if = (
      (    0.0, 0.0 )
      ( corner, 1.0 )
      (    1.0, 1.0 )
   );

   area_ratio = ratio_if;

   int_area_ratio = (
      ( 0.00, 0.00 )
      ( 0.10, 0.50 )
      ( 0.25, 1.00 )
      ( 1.00, 1.00 )
   );

   curvature_ratio = ratio_if;

   complexity_ratio = ratio_if;
```

(continues on next page)

(continued from previous page)

```
   inten_perc_ratio = ratio_if;
}

////////////////////////////////////////////////////////////////////////////////

//
// Total interest threshold for determining matches
//
//total_interest_thresh =
${METPLUS_TOTAL_INTEREST_THRESH}

//
// Interest threshold for printing output pair information
//
print_interest_thresh = 0.0;

////////////////////////////////////////////////////////////////////////////////

//
// Plotting information
//
met_data_dir = "MET_BASE";

fcst_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

obs_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

object_plot = {
   color_table      = "MET_BASE/colortables/mode_obj.ctable";
}

//
// Boolean for plotting on the region of valid data within the domain
//
plot_valid_flag = FALSE;

//
```

(continues on next page)

(continued from previous page)

```
// Plot polyline edges using great circle arcs instead of straight lines
//
plot_gcarc_flag = FALSE;

////////////////////////////////////////////////////////////////////////////////

//
// NetCDF matched pairs, PostScript, and contingency table output files
//
//ps_plot_flag =
${METPLUS_PS_PLOT_FLAG}

//nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

//ct_stats_flag =
${METPLUS_CT_STATS_FLAG}


////////////////////////////////////////////////////////////////////////////////

shift_right = 0;   //  grid squares

////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in MODE_fcstFV3_obsGOES_BrightnessTemp.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/MODE_
↪fcstFV3_obsGOES_BrightnessTemp.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in MODE_fcstFV3_obsGOES_BrightnessTemp.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/MODE_
↪fcstFV3_obsGOES_BrightnessTemp.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in short_range/brightness_temperature (relative to **OUTPUT_BASE**) and will contain the following files:

mode_FV3_core_lsm1_010000L_20190521_010000V_NAA_cts.txt mode_FV3_core_lsm1_010000L_20190521_01000
mode_FV3_core_lsm1_010000L_20190521_010000V_NAA_obj.txt mode_FV3_core_lsm1_010000L_20190521_01000
mode_FV3_core_lsm1_010000L_20190521_020000V_NAA_cts.txt mode_FV3_core_lsm1_010000L_20190521_02000
mode_FV3_core_lsm1_010000L_20190521_020000V_NAA_obj.txt mode_FV3_core_lsm1_010000L_20190521_02000
mode_FV3_core_mp1_010000L_20190521_010000V_NAA_cts.txt mode_FV3_core_mp1_010000L_20190521_01000
mode_FV3_core_mp1_010000L_20190521_010000V_NAA_obj.txt mode_FV3_core_mp1_010000L_20190521_01000
mode_FV3_core_mp1_010000L_20190521_020000V_NAA_cts.txt mode_FV3_core_mp1_010000L_20190521_02000
mode_FV3_core_mp1_010000L_20190521_020000V_NAA_obj.txt mode_FV3_core_mp1_010000L_20190521_02000

Keywords

Note:

- MODEToolUseCase
- MODEToolUseCase
- ShortRangeAppUseCase
- NetCDFFileUseCase
- NOAAEMCOrgUseCase
- NOAAHWTOrgUseCase
- ValidationUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-MODE_fcstFV3_obsGOES_BrightnessTemp.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.12 MODE/Grid-Stat: Brightness Temperature Verification and Distance Maps

model_applications/ short_range/ MODE_fcstFV3_obsGOES_BrightnessTempObjs.conf

Scientific Objective

To provide statistical inforation on regions of low brightness temperatures, defined by creating objects, in the FV3 ensemble members compared to GOES satellite. In addition, distance map information is computed for both the model and observation using object based brightness temperatures

Datasets

- Forecast dataset: FV3 Model member data
- Observation dataset: GOES Brightness Temperature

METplus Components

This use case runs MODE to create object statistics on brightness temperatures below 235 K. Then it runs grid_stat to compute neighborhood contingency table counts and distance maps for the forecast and observations.

METplus Workflow

The MODE and grid_stat tools are run for each of 2 ensemble members and for each time. This example loops by initialization time. It processes 2 lead times, listed below.

Valid: 2019-05-21_01Z
Forecast lead: 01

Valid: 2019-05-21_02Z
Forecast lead: 02

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/short_range/MODE_fcstFV3_obsGOES_BrightnessTempObjs.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/MODE_
↪fcstFV3_obsGOES_BrightnessTempObjs.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = MODE(lsm1), MODE(mp1), GridStat(lsm1), GridStat(mp1)


###
# Time Info
```

(continues on next page)

(continued from previous page)

```
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = init
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2019052100
INIT_END = 2019052100
INIT_INCREMENT = 3600

LEAD_SEQ = 1,2


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

# MODE

FCST_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/brightness_temperature
FCST_MODE_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/core_{instance}/core_{instance}_{init?fmt=%Y%m
↪%d}_{init?fmt=%H%M}_f{lead?fmt=%HH}.nc

OBS_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/brightness_temperature
OBS_MODE_INPUT_TEMPLATE = {valid?fmt=%Y_%m_%d}_141/remap_GOES-16.{valid?fmt=%Y%m%d}.{valid?
↪fmt=%H%M%S}.nc

MODE_OUTPUT_DIR = {OUTPUT_BASE}/short_range/brightness_temperature/mode

# GridStat

FCST_GRID_STAT_INPUT_DIR = {MODE_OUTPUT_DIR}
FCST_GRID_STAT_INPUT_TEMPLATE = mode_{MODE_OUTPUT_PREFIX}_{lead?fmt=%HH}0000L_{valid?fmt=%Y%m
↪%d}_{valid?fmt=%H%M%S}V_000000A_obj.nc

OBS_GRID_STAT_INPUT_DIR = {FCST_GRID_STAT_INPUT_DIR}
OBS_GRID_STAT_INPUT_TEMPLATE = {FCST_GRID_STAT_INPUT_TEMPLATE}
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/short_range/brightness_temperature/grid_stat_obj


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = FV3_core
OBTYPE = GOES

FCST_MODE_VAR1_NAME = SBTA1613_topofatmosphere
FCST_MODE_VAR1_LEVELS = "(*,*)"
FCST_MODE_VAR1_OPTIONS = file_type = NETCDF_MET;
MODE_FCST_CENSOR_THRESH = <=0
MODE_FCST_CENSOR_VAL = 9999

OBS_MODE_VAR1_NAME = channel_13_brightness_temperature
OBS_MODE_VAR1_LEVELS = "(*,*)"
OBS_MODE_VAR1_OPTIONS = file_type = NETCDF_MET;
MODE_OBS_CENSOR_THRESH = <=0
MODE_OBS_CENSOR_VAL = 9999


FCST_GRID_STAT_VAR1_NAME = fcst_obj_raw
FCST_GRID_STAT_VAR1_LEVELS = "(*,*)"
FCST_GRID_STAT_VAR1_THRESH = lt999
FCST_GRID_STAT_VAR1_OPTIONS = file_type = NETCDF_MET; censor_thresh = eq-9999; censor_val =␣
↪999;

OBS_GRID_STAT_VAR1_NAME = obs_obj_raw
OBS_GRID_STAT_VAR1_LEVELS = "(*,*)"
OBS_GRID_STAT_VAR1_THRESH = lt999
OBS_GRID_STAT_VAR1_OPTIONS = file_type = NETCDF_MET; censor_thresh = eq-9999; censor_val =␣
↪999;


###
# MODE Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#mode
###

MODE_QUILT = FALSE
```

(continues on next page)

(continued from previous page)

```
MODE_CONV_RADIUS = 5

MODE_CONV_THRESH = <=235

MODE_MERGE_THRESH = <=235

MODE_MERGE_FLAG = NONE

MODE_MASK_POLY = {INPUT_BASE}/model_applications/short_range/brightness_temperature/CentUS.nc

MODE_GRID_RES = 3

MODE_MAX_CENTROID_DIST = 600.0/grid_res

MODE_INTEREST_FUNCTION_CENTROID_DIST = ( ( 0.0, 1.0 ) (  60.0/grid_res, 1.0 ) ( 450.0/grid_
↪res, 0.0 ) )

MODE_MASK_MISSING_FLAG = BOTH

MODE_MASK_POLY_FLAG = BOTH

MODE_WEIGHT_CENTROID_DIST = 4.0
MODE_WEIGHT_BOUNDARY_DIST = 3.0
MODE_WEIGHT_CONVEX_HULL_DIST = 1.0
MODE_WEIGHT_AREA_RATIO = 4.0
MODE_WEIGHT_INT_AREA_RATIO = 3.0

MODE_TOTAL_INTEREST_THRESH = 0.65

MODE_NC_PAIRS_FLAG_POLYLINES = False

MODE_REGRID_TO_GRID = NONE

MODE_OUTPUT_PREFIX = FV3_core_{instance}


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_OUTPUT_FLAG_NBRCTC = BOTH
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_OUTPUT_FLAG_DMAP = BOTH

GRID_STAT_NC_PAIRS_FLAG_DISTANCE_MAP = TRUE

GRID_STAT_OUTPUT_PREFIX = FV3_core_{instance}
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *MODE MET Configuration* (page 174) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Approximate grid resolution (km)
//
// grid_res =
${METPLUS_GRID_RES}

////////////////////////////////////////////////////////////////////////////////

//
// Run all permutations of radius and threshold
//
// quilt =
${METPLUS_QUILT}

//
// MODE Multivar boolean combination logic
//
//multivar_logic =
${METPLUS_MULTIVAR_LOGIC}

//multivar_intensity_flag =
${METPLUS_MULTIVAR_INTENSITY_FLAG}

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FIELD}

   ${METPLUS_FCST_CENSOR_THRESH}
   ${METPLUS_FCST_CENSOR_VAL}
   ${METPLUS_FCST_CONV_RADIUS}
   ${METPLUS_FCST_CONV_THRESH}
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_FCST_VLD_THRESH}
   ${METPLUS_FCST_FILTER_ATTR_NAME}
   ${METPLUS_FCST_FILTER_ATTR_THRESH}
   ${METPLUS_FCST_MERGE_THRESH}
   ${METPLUS_FCST_MERGE_FLAG}
   ${METPLUS_FCST_FILE_TYPE}
   ${METPLUS_FCST_MULTIVAR_NAME}
   ${METPLUS_FCST_MULTIVAR_LEVEL}
}

obs = {
   ${METPLUS_OBS_FIELD}

   ${METPLUS_OBS_CENSOR_THRESH}
   ${METPLUS_OBS_CENSOR_VAL}
   ${METPLUS_OBS_CONV_RADIUS}
   ${METPLUS_OBS_CONV_THRESH}
   ${METPLUS_OBS_VLD_THRESH}
   ${METPLUS_OBS_FILTER_ATTR_NAME}
   ${METPLUS_OBS_FILTER_ATTR_THRESH}
   ${METPLUS_OBS_MERGE_THRESH}
   ${METPLUS_OBS_MERGE_FLAG}
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_MULTIVAR_NAME}
   ${METPLUS_OBS_MULTIVAR_LEVEL}
}

////////////////////////////////////////////////////////////////////////////////

//
// Handle missing data
//
// mask_missing_flag =
${METPLUS_MASK_MISSING_FLAG}

//
// Match objects between the forecast and observation fields
//
//match_flag =
${METPLUS_MATCH_FLAG}

//
// Maximum centroid distance for objects to be compared
//
//max_centroid_dist =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_MAX_CENTROID_DIST}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine weights
//
//weight = {
${METPLUS_WEIGHT_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Fuzzy engine interest functions
//
interest_function = {

   ${METPLUS_INTEREST_FUNCTION_CENTROID_DIST}

   ${METPLUS_INTEREST_FUNCTION_BOUNDARY_DIST}

   ${METPLUS_INTEREST_FUNCTION_CONVEX_HULL_DIST}

   angle_diff = (
      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 0.0 )
   );

   aspect_diff = (
      (  0.00, 1.0 )
      (  0.10, 1.0 )
      (  0.75, 0.0 )
   );

   corner   = 0.8;
   ratio_if = (
```

(continues on next page)

(continued from previous page)

```
      (    0.0, 0.0 )
      ( corner, 1.0 )
      (    1.0, 1.0 )
   );

   area_ratio = ratio_if;

   int_area_ratio = (
      ( 0.00, 0.00 )
      ( 0.10, 0.50 )
      ( 0.25, 1.00 )
      ( 1.00, 1.00 )
   );

   curvature_ratio = ratio_if;

   complexity_ratio = ratio_if;

   inten_perc_ratio = ratio_if;
}

////////////////////////////////////////////////////////////////////////////////

//
// Total interest threshold for determining matches
//
//total_interest_thresh =
${METPLUS_TOTAL_INTEREST_THRESH}

//
// Interest threshold for printing output pair information
//
print_interest_thresh = 0.0;

////////////////////////////////////////////////////////////////////////////////

//
// Plotting information
//
met_data_dir = "MET_BASE";

fcst_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
```

(continues on next page)

(continued from previous page)

```
}

obs_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
}

object_plot = {
   color_table      = "MET_BASE/colortables/mode_obj.ctable";
}

//
// Boolean for plotting on the region of valid data within the domain
//
plot_valid_flag = FALSE;

//
// Plot polyline edges using great circle arcs instead of straight lines
//
plot_gcarc_flag = FALSE;

////////////////////////////////////////////////////////////////////////////////

//
// NetCDF matched pairs, PostScript, and contingency table output files
//
//ps_plot_flag =
${METPLUS_PS_PLOT_FLAG}

//nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

//ct_stats_flag =
${METPLUS_CT_STATS_FLAG}


////////////////////////////////////////////////////////////////////////////////

shift_right = 0;   //  grid squares

////////////////////////////////////////////////////////////////////////////////

${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0";
```

(continues on next page)

(continued from previous page)

```
tmp_dir = "${MET_TMP_DIR}";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}
```

(continues on next page)

(continued from previous page)

```
//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}
```

(continues on next page)

(continued from previous page)

```
//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in MODE_fcstFV3_obsGOES_BrightnessTempObjs.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/MODE_
↪fcstFV3_obsGOES_BrightnessTempObjs.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in MODE_fcstFV3_obsGOES_BrightnessTempObjs.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/MODE_
↪fcstFV3_obsGOES_BrightnessTempObjs.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in short_range/brightness_temperature (relative to **OUTPUT_BASE**) and will contain the following files:

mode/mode_FV3_core_lsm1_010000L_20190521_010000V_NAA_cts.txt mode/mode_FV3_core_lsm1_010000L_2019
mode/mode_FV3_core_lsm1_010000L_20190521_010000V_NAA_obj.txt mode/mode_FV3_core_lsm1_010000L_201
mode/mode_FV3_core_lsm1_010000L_20190521_020000V_NAA_cts.txt mode/mode_FV3_core_lsm1_010000L_2019
mode/mode_FV3_core_lsm1_010000L_20190521_020000V_NAA_obj.txt mode/mode_FV3_core_lsm1_010000L_201
mode/mode_FV3_core_mp1_010000L_20190521_010000V_NAA_cts.txt mode/mode_FV3_core_mp1_010000L_2019
mode/mode_FV3_core_mp1_010000L_20190521_010000V_NAA_obj.txt mode/mode_FV3_core_mp1_010000L_2019
mode/mode_FV3_core_mp1_010000L_20190521_020000V_NAA_cts.txt mode/mode_FV3_core_mp1_010000L_2019
mode/mode_FV3_core_mp1_010000L_20190521_020000V_NAA_obj.txt mode/mode_FV3_core_mp1_010000L_2019
grid_stat_obj/grid_stat_FV3_core_lsm1_000000L_20190521_010000V_dmap.txt
grid_stat_obj/grid_stat_FV3_core_lsm1_000000L_20190521_010000V_nbrctc.txt
grid_stat_obj/grid_stat_FV3_core_lsm1_000000L_20190521_010000V_pairs.nc
grid_stat_obj/grid_stat_FV3_core_lsm1_000000L_20190521_010000V.stat grid_stat_obj/grid_stat_FV3_core_lsm1_00
grid_stat_obj/grid_stat_FV3_core_lsm1_000000L_20190521_020000V_nbrctc.txt
grid_stat_obj/grid_stat_FV3_core_lsm1_000000L_20190521_020000V_pairs.nc
grid_stat_obj/grid_stat_FV3_core_lsm1_000000L_20190521_020000V.stat grid_stat_obj/grid_stat_FV3_core_mp1_00
grid_stat_obj/grid_stat_FV3_core_mp1_000000L_20190521_010000V_nbrctc.txt
grid_stat_obj/grid_stat_FV3_core_mp1_000000L_20190521_010000V_pairs.nc
grid_stat_obj/grid_stat_FV3_core_mp1_000000L_20190521_010000V.stat grid_stat_obj/grid_stat_FV3_core_mp1_000
grid_stat_obj/grid_stat_FV3_core_mp1_000000L_20190521_020000V_nbrctc.txt
grid_stat_obj/grid_stat_FV3_core_mp1_000000L_20190521_020000V_pairs.nc
grid_stat_obj/grid_stat_FV3_core_mp1_000000L_20190521_020000V.stat

Keywords

Note:

- MODEToolUseCase
- GridStatToolUseCase
- ShortRangeAppUseCase
- NetCDFFileUseCase
- NOAAEMCOrgUseCase
- NOAAHWTOrgUseCase
- ValidationUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-MODE_fcstFV3_obsGOES_BrightnessTempObjs.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.13 Grid-Stat: Surrogate Severe and Practically Perfect Probabilistic Evaluation

model_applications/ short_range/ GridStat_fcstHRRR_obsPracPerfect _SurrogateSevereProb.conf

Scientific Objective

To evaluate the surrogate severe forecasts at predicting Severe weather using the (12Z - 12Z) practically perfect storm reports an obtain probabilistic output statistics.

Datasets

- Forecast dataset: HRRR Surrogate Severe Data
- Observation dataset: Practically Perfect from Local Storm Reports

METplus Components

This use case runs grid_stat to create probabilistic statistics on surrogate severe from the HRRR model and Practially Perfect observations computed from local storm reports.

METplus Workflow

The grid_stat tool is run for each time. This example loops by valid time. It processes 1 valid time, listed below.

Valid: 2020-02-06_12Z
Forecast lead: 36

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/short_range/GridStat_fcstHRRR_obsPracPerfect_SurrogateSevere.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/GridStat_
↪fcstHRRR_obsPracPerfect_SurrogateSevereProb.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
```

(continues on next page)

(continued from previous page)

```
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG=2020020612
VALID_END=2020020612
VALID_INCREMENT=86400

INIT_SEQ = 0
LEAD_SEQ_MIN = 36
LEAD_SEQ_MAX = 36


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/surrogate_severe_prac_
↪perfect
FCST_GRID_STAT_INPUT_TEMPLATE = surrogate_severe_{init?fmt=%Y%m%d}_{lead?fmt=%HHH}V_regrid.nc

OBS_GRID_STAT_INPUT_DIR = {FCST_GRID_STAT_INPUT_DIR}
OBS_GRID_STAT_INPUT_TEMPLATE = StormReps_211.{init?fmt=%Y%m%d}.nc

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/short_range/surrogate_severe_prac_
↪perfect/grid_stat/prob


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = HRRR
OBTYPE = PP

FCST_IS_PROB = true
FCST_GRID_STAT_INPUT_DATATYPE = NETCDF
```

(continues on next page)

(continued from previous page)

```
FCST_VAR1_NAME = MXUPHL_prob_75
FCST_VAR1_LEVELS = "(*,*)"
FCST_VAR1_THRESH = ge0.02
FCST_GRID_STAT_PROB_THRESH = ge0.0, ge0.02, ge0.05, ge0.10, ge0.10, ge0.15, ge0.30, ge0.45,␣
↪ge0.60, ge1.0

FCST_VAR2_NAME = MXUPHL_prob_80
FCST_VAR2_LEVELS = {FCST_VAR1_LEVELS}
FCST_VAR2_THRESH = {FCST_VAR1_THRESH}

FCST_VAR3_NAME = MXPHL_prob_85
FCST_VAR3_LEVELS = {FCST_VAR1_LEVELS}
FCST_VAR3_THRESH = {FCST_VAR1_THRESH}

FCST_VAR4_NAME = MXUPHL_prob_90
FCST_VAR4_LEVELS = {FCST_VAR1_LEVELS}
FCST_VAR4_THRESH = {FCST_VAR1_THRESH}

FCST_VAR5_NAME = MXUPHL_prob_95
FCST_VAR5_LEVELS = {FCST_VAR1_LEVELS}
FCST_VAR5_THRESH = {FCST_VAR1_THRESH}

OBS_VAR1_NAME = Fscale_mask
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_THRESH = ge1.0

OBS_VAR2_NAME = {OBS_VAR1_NAME}
OBS_VAR2_LEVELS = {OBS_VAR1_LEVELS}
OBS_VAR2_THRESH = {OBS_VAR1_THRESH}

OBS_VAR3_NAME = {OBS_VAR1_NAME}
OBS_VAR3_LEVELS = {OBS_VAR1_LEVELS}
OBS_VAR3_THRESH = {OBS_VAR1_THRESH}

OBS_VAR4_NAME = {OBS_VAR1_NAME}
OBS_VAR4_LEVELS = {OBS_VAR1_LEVELS}
OBS_VAR4_THRESH = {OBS_VAR1_THRESH}

OBS_VAR5_NAME = {OBS_VAR1_NAME}
OBS_VAR5_LEVELS = {OBS_VAR1_LEVELS}
OBS_VAR5_THRESH = {OBS_VAR1_THRESH}


###
# GridStat Settings
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_OUTPUT_FLAG_PCT = BOTH
GRID_STAT_OUTPUT_FLAG_PSTD = BOTH
GRID_STAT_OUTPUT_FLAG_PJC = BOTH
GRID_STAT_OUTPUT_FLAG_PRC = BOTH

GRID_STAT_NC_PAIRS_FLAG_LATLON = FALSE
GRID_STAT_NC_PAIRS_FLAG_RAW = FALSE
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
```

(continues on next page)

(continued from previous page)

```
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
```

(continues on next page)

(continued from previous page)

```
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
```

(continues on next page)

(continued from previous page)

```
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstHRRR_obsPracPerfect_SurrogateSevere.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/
↪GridStat_fcstHRRR_obsPracPerfect_SurrogateSevereProb.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstHRRR_obsPracPerfect_SurrogateSevere.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/short_range/
↪GridStat_fcstHRRR_obsPracPerfect_SurrogateSevereProb.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/short_range/surrogate_severe_prac_perfect/grid_stat/prob (relative to **OUTPUT_BASE**) and will contain the following files:

grid_stat_360000L_20200206_120000V_pct.txt
grid_stat_360000L_20200206_120000V_pjc.txt
grid_stat_360000L_20200206_120000V_prc.txt
grid_stat_360000L_20200206_120000V_pstd.txt
grid_stat_360000L_20200206_120000V.stat

Keywords

Note:

- GridStatToolUseCase
- ShortRangeAppUseCase
- NetCDFFileUseCase
- NOAAHWTOrgUseCase
- NCAROrgUseCase
- NOAAHMTOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/short_range-SS_PP_prob.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.13.14 Surrogate Severe Calculation: PCPCombine, GenEnsProd, and RegridDataPlane

model_applications/ short_range/ GenEnsProd_fcstHRRR_fcstOnly _SurrogateSevere.conf

Scientific Objective

Run PCPCombine, GenEnsProd, and RegridDataPlane tools to create surrogate severe probability forecasts (SSPFs) for a given date. SSPFs are a severe weather forecasting tool and is a technique used by the Storm Prediction Center (SPC) as well as others. SSPFs are based on updraft helicity (UH; UH $= \int_{z_0}^{z_t}(\omega * \zeta)dz$) since certain thresholds of UH have been shown as good proxies for severe weather. SSPFs can be thought of as the perfect model forecast. They are derived as follows:

1. Regrid the maximum UH value over the 2-5km layer at each grid point to the NCEP 211 grid (dx = ~80km).
2. Create a binary mask of points that meet a given threshold of UH.
3. Convert the binary mask into a probability field by applying a Gaussian filter.

For more information, please reference Sobash et al. 2011 (https://journals.ametsoc.org/doi/full/10.1175/WAF-D-10-05046.1).

Datasets

There are two dates that can be used as input data for this use case 20190518 or 20200205.

- Input Data: HRRR data - There should 24 grib2 files. - Variable of interest: MXUPHL; the maximum updraft helicity - Level: Z2000-5000; from 2 - 5km - Format: grib2 - Projection: Lambert Conformal
- Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
- Data Source: Originally received from Burkely Gallo at the Storm Prediction Center.

METplus Components

This use case runs the PCPCombine, GenEnsProd, and RegridDataPlane MET tools.

METplus Workflow

This workflow loops over the data by process, meaning that each MET tool will run over all times before moving onto the tool. PCPCombine is called first, followed by GenEnsProd, and then, finally, RegridDataPlane.

METplus Configuration

METplus first loads all of the configurations found in parm/metplus_config. Then it loads any configuration files passed to METplus by the command line.

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/short_range/
↪GenEnsProd_fcstHRRR_fcstOnly_SurrogateSevere.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = PCPCombine, GenEnsProd, RegridDataPlane
```

(continues on next page)

(continued from previous page)

```
###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2020020500
INIT_END=2020020500
INIT_INCREMENT=86400

LEAD_SEQ = 36


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_PCP_COMBINE_RUN = True
FCST_PCP_COMBINE_METHOD = DERIVE
FCST_PCP_COMBINE_STAT_LIST = MAX

FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/model_applications/short_range/surrogate_severe_
↪calc
FCST_PCP_COMBINE_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/hrrr_ncep_{init?fmt=%Y%m%d%H}f{lead?fmt=
↪%HHH}.grib2

FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/short_range/surrogate_severe_calc
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d}/hrrr_ncep_{init?fmt=%Y%m%d%H}f{lead?fmt=
↪%HHH}.nc


GEN_ENS_PROD_INPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
GEN_ENS_PROD_INPUT_TEMPLATE = {FCST_PCP_COMBINE_OUTPUT_TEMPLATE}

GEN_ENS_PROD_OUTPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
```

(continues on next page)

(continued from previous page)

```
GEN_ENS_PROD_OUTPUT_TEMPLATE = gen_ens_prod_{valid?fmt=%Y%m%d}_120000V_ens.nc

FCST_REGRID_DATA_PLANE_RUN = True

FCST_REGRID_DATA_PLANE_INPUT_DIR = {GEN_ENS_PROD_OUTPUT_DIR}
FCST_REGRID_DATA_PLANE_INPUT_TEMPLATE = {GEN_ENS_PROD_OUTPUT_TEMPLATE}

FCST_REGRID_DATA_PLANE_OUTPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
FCST_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = surrogate_severe_{init?fmt=%Y%m%d}_{lead?fmt=%HHH}V_
↪regrid.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = FCST_ens
OBTYPE = ANALYS

ENS_VAR1_NAME = {FCST_PCP_COMBINE_OUTPUT_NAME}
ENS_VAR1_LEVELS = "(*,*)"
ENS_VAR1_THRESH = >=14.2, >=19.0, >=26.0, >=38.0, >=61.0


FCST_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = MXUPHL_24_A1_ENS_FREQ_ge14.2
FCST_REGRID_DATA_PLANE_VAR2_INPUT_FIELD_NAME = MXUPHL_24_A1_ENS_FREQ_ge19.0
FCST_REGRID_DATA_PLANE_VAR3_INPUT_FIELD_NAME = MXUPHL_24_A1_ENS_FREQ_ge26.0
FCST_REGRID_DATA_PLANE_VAR4_INPUT_FIELD_NAME = MXUPHL_24_A1_ENS_FREQ_ge38.0
FCST_REGRID_DATA_PLANE_VAR5_INPUT_FIELD_NAME = MXUPHL_24_A1_ENS_FREQ_ge61.0

FCST_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = "(*,*)"
FCST_REGRID_DATA_PLANE_VAR2_INPUT_LEVEL = "(*,*)"
FCST_REGRID_DATA_PLANE_VAR3_INPUT_LEVEL = "(*,*)"
FCST_REGRID_DATA_PLANE_VAR4_INPUT_LEVEL = "(*,*)"
FCST_REGRID_DATA_PLANE_VAR5_INPUT_LEVEL = "(*,*)"

FCST_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = MXUPHL_prob_75
FCST_REGRID_DATA_PLANE_VAR2_OUTPUT_FIELD_NAME = MXUPHL_prob_80
FCST_REGRID_DATA_PLANE_VAR3_OUTPUT_FIELD_NAME = MXUPHL_prob_85
FCST_REGRID_DATA_PLANE_VAR4_OUTPUT_FIELD_NAME = MXUPHL_prob_90
FCST_REGRID_DATA_PLANE_VAR5_OUTPUT_FIELD_NAME = MXUPHL_prob_95


###
```

(continues on next page)

(continued from previous page)

```
# PCPCombine Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pcpcombine
###

FCST_PCP_COMBINE_INPUT_ACCUMS = 1
FCST_PCP_COMBINE_INPUT_NAMES = MXUPHL
FCST_PCP_COMBINE_INPUT_LEVELS = Z2000-5000
FCST_PCP_COMBINE_OUTPUT_NAME = MXUPHL_24
FCST_PCP_COMBINE_OUTPUT_ACCUM = 24
FCST_PCP_COMBINE_DERIVE_LOOKBACK = 24
FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB


###
# GenEnsProd Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#genensprod
###

GEN_ENS_PROD_N_MEMBERS = 1
GEN_ENS_PROD_ENS_THRESH = 1.0

GEN_ENS_PROD_REGRID_TO_GRID = G211
GEN_ENS_PROD_REGRID_METHOD = MAX
GEN_ENS_PROD_REGRID_WIDTH = 27
GEN_ENS_PROD_REGRID_VLD_THRESH = 0.0
GEN_ENS_PROD_REGRID_CENSOR_THRESH = ==-9999
GEN_ENS_PROD_REGRID_CENSOR_VAL = 0.0

GEN_ENS_PROD_CENSOR_THRESH = ==-9999
GEN_ENS_PROD_CENSOR_VAL = 0.0

GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_MINUS = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_PLUS = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_MIN = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_MAX = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_RANGE = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_VLD_COUNT = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY = TRUE
GEN_ENS_PROD_ENSEMBLE_FLAG_NEP = FALSE
GEN_ENS_PROD_ENSEMBLE_FLAG_NMEP = FALSE
```

(continues on next page)

(continued from previous page)

```
###
# RegridDataPlane Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#regriddataplane
###

REGRID_DATA_PLANE_ONCE_PER_FIELD = False

REGRID_DATA_PLANE_VERIF_GRID = G211

REGRID_DATA_PLANE_METHOD = MAXGAUSS

REGRID_DATA_PLANE_WIDTH = 1

REGRID_DATA_PLANE_GAUSSIAN_DX = 81.271
REGRID_DATA_PLANE_GAUSSIAN_RADIUS = 120
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GenEnsProd MET Configuration* (page 121) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Gen-Ens-Prod configuration file.
//
// For additional information, please see the MET Users Guide.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
//model =
${METPLUS_MODEL}
```

(continues on next page)

(continued from previous page)

```
//
// Output description to be written
// May be set separately in each "obs.field" entry
//
//desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
// May be set separately in each "field" entry
//
//regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
//censor_thresh =
${METPLUS_CENSOR_THRESH}

//censor_val    =
${METPLUS_CENSOR_VAL}

//normalize =
${METPLUS_NORMALIZE}

//cat_thresh    =
${METPLUS_CAT_THRESH}

//nc_var_str    =
${METPLUS_NC_VAR_STR}

//
// Ensemble fields to be processed
//
ens = {
   //file_type =
   ${METPLUS_ENS_FILE_TYPE}

   //ens_thresh =
   ${METPLUS_ENS_THRESH}
```

(continues on next page)

(continued from previous page)

```
   //vld_thresh =
   ${METPLUS_VLD_THRESH}

   //field =
   ${METPLUS_ENS_FIELD}

}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}


////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood ensemble probabilities
//
//nbrhd_prob = {
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
//nmep_smooth = {
${METPLUS_NMEP_SMOOTH_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}

//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Ensemble product output types
```

(continues on next page)

(continued from previous page)

```
// May be set separately in each "ens.field" entry
//
//ensemble_flag = {
${METPLUS_ENSEMBLE_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////

//version = "V10.1.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

The command to run this use case is:

```
run_metplus.py /path/to/METplus/parm/use_cases/model_applications/short_range/GenEnsProd_
↪fcstHRRR_fcstOnly_SurrogateSevere.conf
```

Expected Output

```
# A successful run of this use case will output the following to the screen and logfile::
#
#    INFO: METplus has successfully finished runing.
#
# A successful run will have the following output files in the location defined by {OUTPUT_
↪BASE}, which
# is located in the metplus_system.conf configuration file located in /path/to/METplus/parm/
↪metplus_config.
# This list of files should be found for every time run through METplus. Using the output␣
↪for 20190518 as an example.
#
# **PCPCombine output**:
#
# * 20190518/hrrr_ncep_2019051800f036.nc
#
# **GenEnsProd output**:
#
# * gen_ens_prod_20190519_120000V_ens.nc
#
```

(continues on next page)

(continued from previous page)

```
# **RegridDataPlane output**:
#
# * surrogate_severe_20190518_036V_regrid.nc
#
```

Keywords

Note:

- PCPCombineUseCase
- GenEnsProdUseCase
- RegridDataPlaneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/short_range-GenEnsProd_fcstHRRR_fcstOnly_SurrogateSevere.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.14 Space Weather

Upper atmosphere and geospace model configurations

7.2.16.14.1 GenVxMask: Solar Altitude

model_applications/space_weather/GenVxMask_fcstGloTEC_solar _altitude.conf

Overview

This use case illustrates the use of the gen_vx_mask tool for the space weather domain. It creates a mask for region where the solar altitude angle is less than 45 degrees (low sun angle or sun below the horizon), only letting data through for the region where the sun is high in the sky (i.e., solar altitude angle greater than 45 degrees).

In this use case, the input data is the GloTEC model run assimilated with COSMIC-1 RO data.

This use case runs gen_vx_mask for a couple forecast times from a space weather event known as the St. Patrick's Day Storm (Mar 17, 2015).

Novel aspects of this use case:

- First example use case to run gen_vx_mask on a space weather model (GloTEC)

- Example of how to run gen_vx_mask on NetCDF input data which do not strictly conform to the Climate Forecasts (CF) conventions
- Example of constructing a mask based on the solar altitude angle.
- Changing the mask condition to solar alt <= 0 will mask out the night region.
- Changing the mask condition to solar alt > 0 will mask the day region.

Background: The solar altitude angle is the angle of the sun relative to the Earth's horizon, and is measured in degrees. The altitude is zero at sunrise and sunset, and can reach a maximum of 90 degrees (directly overhead) at noon at latitudes near the equator. [Source: https://sciencing.com/solar-altitude-23364.html]

Scientific Objective

Creating masking region files to be used by other MET tools. This use case applies a solar altitude mask (solar altitude restriction) to the input grid, creating a separate masked output file for each time level of the input file.

Datasets

Input Grid: GloTEC

Masks: Solar altitude

Location: All of the input data required for this use case can be found in the sample data tarball.
Click here to download:
https://github.com/dtcenter/METplus/releases/download/v3.0/sample_data-space_weather-3.0.tgz
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1877) section for more information.

Data source: NOAA Space Weather Prediction Center (SWPC)
Data contact: Dominic Fuller-Rowell (dominic.fuller-rowell@noaa.gov)

METplus Use Case Contact

Author: Jonathan L. Vigh (National Center for Atmospheric Research / Research Applications Laboratory / Joint Numerical Testbed)
Last modified: 26 May 2020

METplus Components

This use case utilizes the METplus GenVxMask wrapper to generate a command to run the MET tool GenVxMask if all required files are found.

METplus Workflow

GenVxMask is the only tool called in this example. It processes the following run time:

Init: 2015-03-17 0005Z
Forecast lead: 0

Init: 2015-03-17 0015Z
Forecast lead: 0

The input file is read to define the output grid. Then the solar altitude angle specified with the -thresh argument is applied to the input file, creating the output file.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/space_weather/GenVxMask_fcstGloTEC_FcstOnly_solar_altitude.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/space_weather/
↪GenVxMask_fcstGloTEC_FcstOnly_solar_altitude.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GenVxMask


###
```

(continues on next page)

(continued from previous page)

```
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

# Just run the first two time points for this use case example
# replace with 201503172355 process the entire day

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H%M
VALID_BEG = 201503170005
VALID_END = 201503170015
VALID_INCREMENT = 600

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

GEN_VX_MASK_INPUT_DIR = {INPUT_BASE}/model_applications/space_weather/glotec_vs_glotec/GLO_
↪20190422_with_cosmic
GEN_VX_MASK_INPUT_TEMPLATE = GloTEC_TEC_{valid?fmt=%Y_%m_%d}_cosmic.nc

GEN_VX_MASK_INPUT_MASK_DIR = {INPUT_BASE}/model_applications/space_weather/glotec_vs_glotec/
↪GLO_20190422_with_cosmic
GEN_VX_MASK_INPUT_MASK_TEMPLATE = GloTEC_TEC_{valid?fmt=%Y_%m_%d}_cosmic.nc

GEN_VX_MASK_OUTPUT_DIR={OUTPUT_BASE}/model_applications/space_weather/GenVxMask_glotec_solar_
↪altitude
GEN_VX_MASK_OUTPUT_TEMPLATE = GloTEC_TEC_solar_altitude_le_45_masked_{valid?fmt=%Y_%m_%d_%H
↪%M}.nc


###
# GenVxMask Settings
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#genvxmask
###

GEN_VX_MASK_OPTIONS = -type solar_alt -thresh 'le45' -name TEC_with_solar_altitude_angle_le_
↪45_masked_{valid?fmt=%Y_%m_%d_%H%M} -input_field 'name="TEC"; level="({valid?fmt=%Y%m%d_%H
↪%M%S},*,*)"; file_type=NETCDF_NCCF;' -mask_field 'name="TEC"; level="({valid?fmt=%Y%m%d_%H
↪%M%S},*,*)"; file_type=NETCDF_NCCF;'
```

MET Configuration

None. GenVxMask does not use configuration files.

Running METplus

This use case can be run two ways:

1) Passing in the use case config file then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/space_weather/
↪GenVxMask_fcstGloTEC_FcstOnly_solar_altitude.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in the use case config file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/space_weather/
↪GenVxMask_fcstGloTEC_FcstOnly_solar_altitude.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/space_weather/GenVxMask_fcstGloTEC_solar_altitude (relative to **OUTPUT_BASE**) and will contain the following files:

- GloTEC_TEC_solar_altitude_le_45_masked_2015_03_17_0005.nc
- GloTEC_TEC_solar_altitude_le_45_masked_2015_03_17_0015.nc

Keywords

Note:

- GenVxMaskToolUseCase
- SpaceWeatherAppUseCase
- NOAASWPCOrgUseCase
- MaskingFeatureUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/space_weather-GenVxMask_fcstGloTEC_solar_altitude.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.14.2 Grid-Stat: Analysis validation

GridStat_fcstGloTEC _obsGloTEC_vx7.conf

Overview

This use case illustrates the use of grid_stat tool for the space weather domain. It compares Total Electron Content for a GloTEC model run initialized with COSMIC-1 radio occultation (RO) data to a GloTEC model run without such data.

In this use case, the forecast is considered to be the run without COSMIC-1 RO data. The observations are considered to be the run with COSMIC-1 RO data.

This use case runs grid_stat for the first two forecast times of a space weather event known as the St. Patrick's Day Storm (Mar 17, 2015).

Novel aspects of this use case:

- This is the first example use case to run grid_stat on a space weather model (GloTEC)
- Example of how to run with NetCDF input data which do not strictly conform to the Climate Forecasts (CF) conventions
- Example of using masks covering latitudinal bands of interest to the space weather community: equatorial region, mid-latitude region, and polar region
- Example of masking using the values of a quality flag which vary at each time step and grid point

Scientific Objective

Compare gridded forecast data from a run of the GloTEC model that includes assimilation of COSMIC-1 radio occultation (RO) observations to gridded forecast data from a GloTEC model run that does not include COSMIC-1 RO data.

Datasets

Forecast: GloTEC Total Electron Content (TEC) model run without assimilation of any COSMIC-1 RO data
Observation: GloTEC TEC model run that assimilates COSMIC-1 RO data
Location: Click here for the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1887) section for more information.
Data source: NOAA Space Weather Prediction Center (SWPC)
Data contact: Dominic Fuller-Rowell (dominic.fuller-rowell@noaa.gov)

METplus Use Case Contact

Author: Jonathan L. Vigh (National Center for Atmospheric Research / Research Applications Laboratory / Joint Numerical Testbed)
Last modified: 06 February 2020

METplus Components

This use case utilizes the METplus GridStat wrapper to search for files that are valid at a given run time and generate a command to run the MET tool grid_stat if all required files are found.

METplus Workflow

GridStat is the only tool called in this example. It processes the following run times:

Init: 2015-03-17 0005Z
Forecast lead: 0

Init: 2015-03-17 0015Z
Forecast lead: 0

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/space_weather/GridStat_fcstGloTEC_obsGloTEC_vx7.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/space_weather/
↪GridStat_fcstGloTEC_obsGloTEC_vx7.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
```

(continues on next page)

(continued from previous page)

```
###

# Just run the first two time points for this use case example
# replace with 201503172355 process the entire day

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H%M
VALID_BEG = 201503170005
VALID_END = 201503170015
VALID_INCREMENT = 600

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/space_weather/glotec_vs_glotec/
↪GLO_20190422_without_cosmic
FCST_GRID_STAT_INPUT_TEMPLATE = GloTEC_TEC_{valid?fmt=%Y_%m_%d}.nc

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/space_weather/glotec_vs_glotec/GLO_
↪20190422_with_cosmic
OBS_GRID_STAT_INPUT_TEMPLATE = GloTEC_TEC_{valid?fmt=%Y_%m_%d}_cosmic.nc

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/space_weather/glotec_vs_glotec
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y_%m_%d}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = GloTEC_without_cosmic
OBTYPE = GloTEC_with_cosmic

BOTH_VAR1_NAME = TEC
BOTH_VAR1_LEVELS = "({valid?fmt=%Y%m%d_%H%M%S},*,*)"


###
```

(continues on next page)

(continued from previous page)

```
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_MET_CONFIG_OVERRIDES =  file_type = NETCDF_NCCF;

GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_MCTC = STAT
GRID_STAT_OUTPUT_FLAG_MCTS = STAT
GRID_STAT_OUTPUT_FLAG_CNT = STAT
GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT

GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_ONCE_PER_FIELD = False

GRID_STAT_OUTPUT_PREFIX={MODEL}-vx7_{CURRENT_OBS_NAME}_vs_{OBTYPE}

GRID_STAT_DESC = vx7

# Masking poly for GridStat
MODEL_FILE={FCST_GRID_STAT_INPUT_DIR}/{FCST_GRID_STAT_INPUT_TEMPLATE}
MODEL_LEVEL={valid?fmt=%Y%m%d_%H%M%S},*,*)
MASK_DIR={INPUT_BASE}/model_applications/space_weather/glotec_vs_glotec/masks
GRID_STAT_MASK_POLY = {MODEL_FILE} {name = "quality_flag"; level = "{MODEL_LEVEL}"; file_
↪type=NETCDF_NCCF;} ==0, {MODEL_FILE} {name = "quality_flag"; level = "{MODEL_LEVEL}"; file_
↪type=NETCDF_NCCF;} ==1, {MODEL_FILE} {name = "quality_flag"; level = "{MODEL_LEVEL}"; file_
↪type=NETCDF_NCCF;} ==2, {MODEL_FILE} {name = "quality_flag"; level = "{MODEL_LEVEL}"; file_
↪type=NETCDF_NCCF;} ==3, {MODEL_FILE} {name = "quality_flag"; level = "{MODEL_LEVEL}"; file_
↪type=NETCDF_NCCF;} ==4, {MODEL_FILE} {name = "quality_flag"; level = "{MODEL_LEVEL}"; file_
↪type=NETCDF_NCCF;} ==5, {MASK_DIR}/EQUATORIAL.nc, {MASK_DIR}/MIDLATITUDE.nc, {MASK_DIR}/
↪POLAR.nc
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *GridStat MET Configuration* (page 152) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
```

(continues on next page)

(continued from previous page)

```
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Neighborhood methods
//
nbrhd = {
```

(continues on next page)

(continued from previous page)

```
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstGloTEC_obsGloTEC_vx7.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/space_weather/
↪GridStat_fcstGloTEC_obsGloTEC_vx7.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstGloTEC_obsGloTEC_vx7.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/space_weather/
↪GridStat_fcstGloTEC_obsGloTEC_vx7.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in space_weather/glotec_vs_glotec/output_data/2015_03_17 (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_GloTEC_without_cosmic-vx7_TEC_vs_GloTEC_with_cosmic_000000L_20150317_000500V_pairs.nc
- grid_stat_GloTEC_without_cosmic-vx7_TEC_vs_GloTEC_with_cosmic_000000L_20150317_001500V_pairs.nc
- grid_stat_GloTEC_without_cosmic-vx7_TEC_vs_GloTEC_with_cosmic_000000L_20150317_000500V.stat
- grid_stat_GloTEC_without_cosmic-vx7_TEC_vs_GloTEC_with_cosmic_000000L_20150317_001500V.stat

Keywords

Note:

- GridStatToolUseCase
- SpaceWeatherAppUseCase
- NOAASWPCOrgUseCase
- CustomStringLoopingUseCase
- MaskingFeatureUseCase
- ValidationUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/space_weather-GridStat_fcstGloTEC_obsGloTEC_vx7.jpg'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.15 Tropical Cyclone and Extra Tropical Cyclone

Any field that is associated with Tropical Cyclone and Extra-tropical Cyclones

7.2.16.15.1 TCRMW: Hurricane Gonzalo

model_applications/tc_and_extra_tc/TCRMW_fcstGFS_fcstOnly _gonzolo.conf

Scientific Objective

The TC-RMW tool regrids tropical cyclone model data onto a moving range-azimuth grid centered on points along the storm track. This capability replicates the NOAA Hurricane Research Division DIA-Post module.

Datasets

Forecast: GFS GRIB2
Track: A Deck

Location: All of the input data required for this use case can be found in the tc_and_extra_tc sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1894) section for more information.

METplus Components

This use case utilizes the METplus TCRMW wrapper to search for the desired ADECK file and forecast files that are correspond to the track. It generates a command to run the MET tool TC-RMW if all required files are found.

METplus Workflow

TCRMW is the only tool called in this example. It processes the following run times:

Init: 2014-10-13 12Z
Forecast lead: 0, 6, 12, 18, and 24 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/tc_and_extra_tc/TCRMW_fcstGFS_fcstOnly_gonzalo.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/tc_and_extra_tc/
↪TCRMW_fcstGFS_fcstOnly_gonzalo.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCRMW


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
```

(continues on next page)

(continued from previous page)

```
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2014101312
INIT_END = 2014101312
INIT_INCREMENT = 6H

LEAD_SEQ = begin_end_incr(0, 24, 6)
#LEAD_SEQ = begin_end_incr(0, 126, 6)


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

TC_RMW_DECK_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/rmw/adeck
TC_RMW_DECK_TEMPLATE = gonzalo08l.{init?fmt=%Y%m%d%H}.f00-24.trak.hwrf.atcfunix.06hr

TC_RMW_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/rmw/fcst
TC_RMW_INPUT_TEMPLATE = gonzalo08l.subset.{init?fmt=%Y%m%d%H}.hwrfprs.core.0p02.f{lead?fmt=
↪%3H}.grb2

TC_RMW_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/tc_and_extra_tc/TCRMW_gonzalo
TC_RMW_OUTPUT_TEMPLATE = tc_rmw_gonzal09l.{init?fmt=%Y%m%d%H}.nc


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = HCLT

PRESSURE_LEVELS = "P1000","P850","P700","P500","P300","P200","P150","P100"

BOTH_VAR1_NAME = PRMSL
BOTH_VAR1_LEVELS = L0

BOTH_VAR2_NAME = PRES
BOTH_VAR2_LEVELS = L0

BOTH_VAR3_NAME = TMP
BOTH_VAR3_LEVELS = {PRESSURE_LEVELS}

BOTH_VAR4_NAME = RH
```

(continues on next page)

(continued from previous page)

```
BOTH_VAR4_LEVELS = {PRESSURE_LEVELS}

BOTH_VAR5_NAME = UGRD
BOTH_VAR5_LEVELS = {PRESSURE_LEVELS}

BOTH_VAR5_NAME = VGRD
BOTH_VAR5_LEVELS = {PRESSURE_LEVELS}


###
# TCRMW Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcrmw
###

TC_RMW_INPUT_DATATYPE = GRIB2

TC_RMW_REGRID_METHOD = BILIN
TC_RMW_REGRID_WIDTH = 2
TC_RMW_REGRID_VLD_THRESH = 0.5
TC_RMW_REGRID_SHAPE = SQUARE
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TCRMW MET Configuration* (page 290) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// TC-RMW configuration file.
//
// For additional information, see the MET_BASE/config/README_TC file.
//
////////////////////////////////////////////////////////////////////////////////

// The following environment variables set the text if the corresponding
```

(continues on next page)

(continued from previous page)

```
// variables at defined in the METplus config. If not, they are set to
// and empty string, which will cause MET to use the value defined in the
// default configuration file.

${METPLUS_MODEL}

${METPLUS_STORM_ID}
${METPLUS_BASIN}
${METPLUS_CYCLONE}
${METPLUS_INIT_INCLUDE}

${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INCLUDE_LIST}
${METPLUS_VALID_EXCLUDE_LIST}

${METPLUS_VALID_HOUR_LIST}
${METPLUS_LEAD_LIST}

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];

//
// Data fields
//
data = {
   ${METPLUS_DATA_FILE_TYPE}

   ${METPLUS_DATA_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Regridding options
//
${METPLUS_REGRID_DICT}

//
// Range-Azimuth grid parameters
//
// The following environmnet variables set the text if the corresponding
```

(continues on next page)

(continued from previous page)

```
// variables at defined in the METplus config. If not, they are set to
// and empty string, which will cause MET to use the value defined in the
// default configuration file.

${METPLUS_N_RANGE}
${METPLUS_N_AZIMUTH}
${METPLUS_MAX_RANGE_KM}
${METPLUS_DELTA_RANGE_KM}
${METPLUS_RMW_SCALE}

////////////////////////////////////////////////////////////////////////////////

//version = "V10.0";

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in the use case configuration file then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/model_applications/tc_and_extra_tc/TCRMW_
↪fcstGFS_fcstOnly_gonzalo.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in use case configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/tc_and_extra_tc/
↪TCRMW_fcstGFS_fcstOnly_gonzalo.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/tc_and_extra_tc/TCRMW_gonzalo (relative to **OUTPUT_BASE**) and will contain the following files:

- tc_rmw_aal142016.nc

Keywords

Note:

- TCRMWToolUseCase
- GRIB2FileUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/tc_and_extra_tc-TCRMW_fcstGFS_fcstOnly_gonzolo.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.15.2 Track and Intensity Plotter: Generate mean, median and box plots

model_applications/tc_and_extra_tc/Plotter_fcstGFS _obsGFS_RPlotting.conf

Scientific Objective

By maintaining focus of each evaluation time on a user-defined area around a cyclone, the model statistical errors associated with cyclonic physical features (moisture flux, stability, strength of upper-level PV anomaly and jet, etc.) can be related directly to the model forecasts and provide improvement guidance by accurately depicting interactions with significant weather features around and within the cyclone. This is in contrast to the traditional method of regional averaging cyclone observations in a fixed grid, which "smooths out" system features and limits the meaningful metrics that can be gathered. This use case relays the mean and median of forecast lead times for cyclone position compared to a reference dataset via boxplot.

Datasets

- Forecast dataset: ADeck ATCF tropical cyclone data
- Observation dataset: BDeck ATCF tropical cyclone "best track" cyclone data

METplus Components

This use case first runs TCPairs and then generates the requested plot types for statistics of interest. The TCMPRPlotterConfig_customize configuration file is used by the plot_tcmpr.R script to select things such as the size of the plot window that appears on your screen, etc.

METplus Workflow

The following tools are used for each run time:

TCPairs > plot_tcmpr.R

To generate TCPairs output, this example loops by initialization time for every 6 hour period that is available in the data set for 20141214. The output is then used to generate the mean, median, and box plot for the following: the difference between the MSLP of the Adeck and Bdeck tracks (AMSLP-BMSLP), the difference between the max wind of the Adeck and Bdeck tracks (AMAX_WIND-BMSLP), and the track err (TK_ERR).

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/tc_and_extra_tc/Plotter_fcstGFS_obsGFS_RPlotting.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/tc_and_extra_tc/
↪Plotter_fcstGFS_obsGFS_RPlotting.html

```

(continues on next page)

(continued from previous page)

```
# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCPairs, TCMPRPlotter


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20141214
INIT_END = 20141214
INIT_INCREMENT = 21600 ;; set to every 6 hours=21600 seconds


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

# TCPairs

TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/track_data
TC_PAIRS_ADECK_TEMPLATE = {date?fmt=%Y%m}/a{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}

TC_PAIRS_BDECK_INPUT_DIR = {TC_PAIRS_ADECK_INPUT_DIR}
TC_PAIRS_BDECK_TEMPLATE = {date?fmt=%Y%m}/b{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}
```

(continues on next page)

(continued from previous page)

```
TC_PAIRS_REFORMAT_DIR = {OUTPUT_BASE}/track_data_atcf
TC_PAIRS_SKIP_IF_REFORMAT_EXISTS = yes

TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs
TC_PAIRS_OUTPUT_TEMPLATE = {date?fmt=%Y%m}/{basin?fmt=%s}q{date?fmt=%Y%m%d%H}.gfso.{cyclone?
↪fmt=%s}
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = yes


# TCMPRPlotter

TCMPR_PLOTTER_TCMPR_DATA_DIR = {TC_PAIRS_OUTPUT_DIR}
TCMPR_PLOTTER_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/tcmpr_plots


###
# TCPairs Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcpairs
###

TC_PAIRS_DLAND_FILE = {MET_INSTALL_DIR}/share/met/tc_data/dland_global_tenth_degree.nc

TC_PAIRS_REFORMAT_DECK = yes
TC_PAIRS_REFORMAT_TYPE = SBU

TC_PAIRS_MISSING_VAL_TO_REPLACE = -99
TC_PAIRS_MISSING_VAL = -9999


###
# TCMPRPlotter Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcmprplotter
###

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/tc_and_extra_tc/Plotter_fcstGFS_obsGFS_
↪RPlotting
TCMPR_PLOTTER_CONFIG_FILE = {CONFIG_DIR}/TCMPRPlotterConfig_customize

TCMPR_PLOTTER_PREFIX =
TCMPR_PLOTTER_TITLE =
TCMPR_PLOTTER_SUBTITLE = Your subtitle goes here
TCMPR_PLOTTER_XLAB =
TCMPR_PLOTTER_YLAB = Your y-label goes here
TCMPR_PLOTTER_XLIM =
```

(continues on next page)

(continued from previous page)

```
TCMPR_PLOTTER_YLIM =
TCMPR_PLOTTER_FILTER =
# the tcst data file to be used instead of running the MET tc_stat tool.
TCMPR_PLOTTER_FILTERED_TCST_DATA_FILE =
# Comma separated, no whitespace.  Default is TK_ERR (track error) unless
# otherwise indicated.
TCMPR_PLOTTER_DEP_VARS =AMSLP-BMSLP, AMAX_WIND-BMAX_WIND, TK_ERR
TCMPR_PLOTTER_SCATTER_X =
TCMPR_PLOTTER_SCATTER_Y =
TCMPR_PLOTTER_SKILL_REF =
TCMPR_PLOTTER_SERIES =
TCMPR_PLOTTER_SERIES_CI =
TCMPR_PLOTTER_LEGEND = Your legend text goes here...
TCMPR_PLOTTER_LEAD =
# Mean and median plots. These override the plot_tcmpr.R default of box plot.
# If box plot is desired, this needs to be explicitly indicated.
TCMPR_PLOTTER_PLOT_TYPES = MEAN,MEDIAN,BOXPLOT
TCMPR_PLOTTER_RP_DIFF =
TCMPR_PLOTTER_DEMO_YR =
TCMPR_PLOTTER_HFIP_BASELINE =
TCMPR_PLOTTER_FOOTNOTE_FLAG =
TCMPR_PLOTTER_PLOT_CONFIG_OPTS =
TCMPR_PLOTTER_SAVE_DATA =

#  TCMPR FLAGS no == (don't set flag), yes == (set flag)
TCMPR_PLOTTER_NO_EE = no
TCMPR_PLOTTER_NO_LOG = no
TCMPR_PLOTTER_SAVE = no
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TCPairs MET Configuration* (page 282) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

//
// Models
//
${METPLUS_MODEL}

//
// Description
//
${METPLUS_DESC}

//
// Storm identifiers
//
${METPLUS_STORM_ID}

//
// Basins
//
${METPLUS_BASIN}

//
// Cyclone numbers
//
${METPLUS_CYCLONE}

//
// Storm names
//
${METPLUS_STORM_NAME}

//
// Model initialization time windows to include or exclude
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
```

(continues on next page)

(continued from previous page)

```
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
${METPLUS_VALID_INC}
// valid_exc =
${METPLUS_VALID_EXC}

// write_valid =
${METPLUS_WRITE_VALID}

//
// Valid model time window
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}

//
// Model initialization hours
//
init_hour = [];

//
// Required lead time in hours
//
lead_req = [];

//
// lat/lon polylines defining masking regions
//
init_mask  = "";
valid_mask = "";

//
// Specify if the code should check for duplicate ATCF lines
//
//check_dup =
${METPLUS_CHECK_DUP}


//
// Specify special processing to be performed for interpolated models.
// Set to NONE, FILL, or REPLACE.
```

(continues on next page)

(continued from previous page)

```
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
//
//consensus =
${METPLUS_CONSENSUS_LIST}


//
// Forecast lag times
//
lag_time = [];

//
// CLIPER/SHIFOR baseline forecasts to be derived from the BEST
// and operational (CARQ) tracks.
//
best_technique = [ "BEST" ];
best_baseline  = [];
oper_technique = [ "CARQ" ];
oper_baseline  = [];

//
// Specify the datasets to be searched for analysis tracks (NONE, ADECK, BDECK,
// or BOTH).
//
anly_track = BDECK;

//
// Specify if only those track points common to both the ADECK and BDECK
// tracks be written out.
//
//match_points =
${METPLUS_MATCH_POINTS}

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
${METPLUS_DLAND_FILE}

//
```

(continues on next page)

(continued from previous page)

```
// Specify watch/warning information:
//   - Input watch/warning filename
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}


//diag_info_map = {
${METPLUS_DIAG_INFO_MAP_LIST}

//diag_convert_map = {
${METPLUS_DIAG_CONVERT_MAP_LIST}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

NOTE - In order for this example to run successfully, ensure that your output folder ({OUTPUT_BASE}/tc_pairs/201412) is empty. If there are any files in this directory, the program will fail out and not produce the output for {OUTPUT_BASE}/tcmpr_plots.

This use case can be run two ways:

1) Passing in Plotter_fcstGFS_obsGFS_RPlotting.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/tc_and_extra_tc/
↪Plotter_fcstGFS_obsGFS_RPlotting.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in Plotter_fcstGFS_obsGFS_RPlotting.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/tc_and_extra_tc/
↪Plotter_fcstGFS_obsGFS_RPlotting.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify

the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. TCPairs output for this use case will be found in tc_pairs/201412 (relative to **OUTPUT_BASE**) and will contain files with the following format:

- mlq2014121400.gfso.<*nnnn*>.tcst

where *nnnn* is a zero-padded 4-digit number

Plots (in .png format) will be found in tcmpr_plots (relative to **OUTPUT_BASE**): * AMAX_WIND-BMAX_WIND_boxplot.png * AMAX_WIND-BMAX_WIND_boxplot.png * AMAX_WIND-BMAX_WIND_boxplot.png * AMSLP-BMSLP_boxplot.png * AMSLP-BMSLP_boxplot.png * AMSLP-BMSLP_boxplot.png * TK_ERR_boxplot.png * TK_ERR_mean.png * TK_ERR_median.png

Keywords

Note:

- TCPairsToolUseCase
- TCandExtraTCAppUseCase
- FeatureRelativeUseCase
- MediumRangeAppUseCase

- SBUOrgUseCase
- DTCOrgUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/tc_and_extra_tc-Plotter_fcstGFS_obsGFS_RPlotting.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.15.3 Grid-Stat: Verification of TC forecasts against merged TDR data

model_applications/tc_and_extra_tc/GridStat_fcstHAFS_obsTDR _NetCDF.conf

Scientific Objective

To provide useful statistical information on the relationship between merged Tail Doppler Radar (TDR) data in NetCDF format to a gridded forecast. These values can be used to assess the skill of the prediction. The TDR data is available every 0.5 km AGL. So, the TC forecasts need to be in height coordinates to compare with the TDR data.

Datasets

Forecast: HAFS zonal wind
Observation: HRD TDR merged_zonal_wind

Location of Model forecast and Dropsonde files: All of the input data required for this use case can be found in the sample data tarball. Click here to download.
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See 'Running METplus' section for more information.

TDR Data Source: Hurricane Research Division: Contact: Paul Reasor Email: paul.reasor@noaa.gov
The data dataset used in the use case is a subset of the Merged Analysis
(v2d_combined_xy_rel_merged_ships.nc).
Thanks to HRD for providing us the dataset

METplus Components

The observations in the use case contains data mapped into Cartesian Grids with a horizontal grid spacing of 2 km and vertical grid spacing of 0.5 km. Hence the model output needs to be in height (km) (vertical coordinates) instead of pressure levels. Both observation and model output are available with the release. The instructions below tells how the input to the use case was prepared. The Hurricane Analysis and Forecast System (HAFS) (pressure levels in GRIB2 format) outputs are converted to height level (in NetCDF4 format) using METcalcpy vertical interpolation routine. Under METcalcpy/examples directory user can modify the vertical_interp_hwrf.sh or create a similar file for their own output. The $DATA_DIR is the top level output directory where the pressure level data resides. The –input and –output should point to the input and output file names resp. The –config points to a yaml file. Users should edit the yaml file, if needed. For this use case only zonal wind (u) at 4 (200m, 2000m, 4000m and 6000m) vertical levels are provided. The use case will compare the HAFS 2 km zonal wind (u) data against TDR's merged_zonal_wind at 2km. The user need to run the shell script to get the height level output in NetCDF4 format. This use case utilizes the METplus python embedding to read the TDR data and compare them to gridded forecast data using GridStat.

METplus Workflow

The use case runs the python embedding scripts (GridStat_fcstHAFS_obsTDR_NetCDF/read_tdr.py: to read the TDR data) and run Grid-Stat (compute statistics against HAFS model output, in height coordinates), called in this example.

It processes the following run times: Valid at 2019-08-29 12Z

Forecast lead times: 0,6,12 and 18 UTC

The mission number (e.g CUSTOM_LOOP_LIST = 190829H1)

Height level (for TDR: OBS_VERT_LEVEL_KM = 2, HAFS: FCST_VAR1_LEVELS = "(0,1,*,*)")

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/tc_and_extra_tc/GridStat_fcstHAFS_obsTDR_NetCDF.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/tc_and_extra_tc/
↪GridStat_fcstHAFS_obsTDR_NetCDF.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = GridStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2019082912
VALID_END = 2019082912
VALID_INCREMENT = 21600

LEAD_SEQ = 0,6,12,18

CUSTOM_LOOP_LIST = 190829H1


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/GridStat_fcstHAFS_
↪obsTDR_NetCDF/hafs_height
FCST_GRID_STAT_INPUT_TEMPLATE = dorian05l.{init?fmt=%Y%m%d%H}.hafsprs.synoptic.0p03.f{lead?
↪fmt=%HHH}.nc4

OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/GridStat_fcstHAFS_
↪obsTDR_NetCDF/obs
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/tc_and_extra_tc/tdr
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

# Location of the TDR file
TC_RADAR_FILE = {OBS_GRID_STAT_INPUT_DIR}/merged_zonal_wind_tdr.nc

# Obs vertical level in km
OBS_VERT_LEVEL_KM = 2

MODEL = HAFS
OBTYPE = TDR

FCST_VAR1_NAME =  u
FCST_VAR1_LEVELS =  "(0,1,*,*)"
FCST_VAR1_THRESH = gt10.0, gt20.0, lt-10.0, lt-20.0
FCST_VAR1_OPTIONS = set_attr_init="{init?fmt=%Y%m%d_%H%M%S}"; set_attr_valid="{valid?fmt=%Y%m
↪%d_%H%M%S}"; set_attr_lead="{lead?fmt=%H}";
FCST_GRID_STAT_INPUT_DATATYPE = NETCDF_NCCF

OBS_VAR1_NAME = {PARM_BASE}/use_cases/model_applications/tc_and_extra_tc/GridStat_fcstHAFS_
↪obsTDR_NetCDF/read_tdr.py {TC_RADAR_FILE} merged_zonal_wind {custom?fmt=%s} {OBS_VERT_
↪LEVEL_KM}
OBS_VAR1_THRESH = gt10.0, gt20.0, lt-10.0, lt-20.0


###
# GridStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#gridstat
###

GRID_STAT_OUTPUT_FLAG_FHO = BOTH
GRID_STAT_OUTPUT_FLAG_CTC = STAT
GRID_STAT_OUTPUT_FLAG_CTS = STAT
GRID_STAT_OUTPUT_FLAG_CNT = STAT
GRID_STAT_OUTPUT_FLAG_SL1L2 = STAT
GRID_STAT_OUTPUT_FLAG_ECLV = NONE

GRID_STAT_REGRID_TO_GRID = OBS

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_ONCE_PER_FIELD = False

GRID_STAT_OUTPUT_PREFIX = {MODEL}_vs_{OBTYPE}
```

MET Configuration

METplus sets environment variables based on the values in the METplus configuration file. These variables are referenced in the MET configuration file. **YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!** If there is a setting in the MET configuration file that is not controlled by an environment variable, you can add additional environment variables to be set only within the METplus environment using the [user_env_vars] section of the METplus configuration files. See the 'User Defined Config' section on the 'System Configuration' page of the METplus User's Guide for more information.

```
////////////////////////////////////////////////////////////////////////////////
//
// Grid-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

//
// Output observation type to be written
//
// obtype =
${METPLUS_OBTYPE}
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//censor_thresh =
${METPLUS_CENSOR_THRESH}
//censor_val =
${METPLUS_CENSOR_VAL}
cat_thresh           = [];
cnt_thresh           = [ NA ];
cnt_logic            = UNION;
wind_thresh          = [ NA ];
wind_logic           = UNION;
eclv_points      = 0.05;
//nc_pairs_var_name =
${METPLUS_NC_PAIRS_VAR_NAME}
nc_pairs_var_suffix = "";
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}

rank_corr_flag   = FALSE;

//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}
obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology mean data
//
```

(continues on next page)

(continued from previous page)

```
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}


//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
// mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// Data smoothing methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Neighborhood methods
//
nbrhd = {
   field      = BOTH;
   // shape =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
   // cov_thresh =
   ${METPLUS_NBRHD_COV_THRESH}
   vld_thresh = 1.0;
}

////////////////////////////////////////////////////////////////////////////////

//
// Fourier decomposition
// May be set separately in each "obs.field" entry
//
//fourier = {
${METPLUS_FOURIER_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Gradient statistics
// May be set separately in each "obs.field" entry
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

////////////////////////////////////////////////////////////////////////////////

//
// Distance Map statistics
// May be set separately in each "obs.field" entry
//
//distance_map = {
${METPLUS_DISTANCE_MAP_DICT}

////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Note the following variables are referenced in the MET configuration file.

Python Embedding

This use case uses a Python embedding script to read input data

parm/use_cases/model_applications/tc_and_extra_tc/GridStat_fcstHAFS_obsTDR_NetCDF/read_tdr.py

```
import os
import sys
```

(continues on next page)

(continued from previous page)

```
sys.path.insert(0, os.path.abspath(os.path.dirname(__file__)))

import tdr_utils

if len(sys.argv) < 5:
    print("Must specify exactly one input file, variable name, mission ID (YYMMDDID), level␣
↪(in km)")
    sys.exit(1)

# Read the input file as the first argument
input_file   = os.path.expandvars(sys.argv[1])
var_name     = sys.argv[2]
mission_name = sys.argv[3]
level_km     = float(sys.argv[4])

met_data, attrs = tdr_utils.main(input_file, var_name, mission_name, level_km)
```

The above script imports another script called tdr_utils.py in the same directory:

parm/use_cases/model_applications/tc_and_extra_tc/GridStat_fcstHAFS_obsTDR_NetCDF/tdr_utils.py

```
from netCDF4 import Dataset
import numpy as np
import datetime as dt
import os
import sys
from time import gmtime, strftime

# Return valid time
def get_valid_time(input_file, mission_name):
    f = Dataset(input_file, 'r')
    mid = f.variables['mission_ID'][:].tolist().index(mission_name)
    valid_time = calculate_valid_time(f, mid)
    valid_time_mid = valid_time.strftime("%Y%m%d%H%M")
    return valid_time_mid

def calculate_valid_time(f, mid):
  merge_year_np  = np.array(f.variables['merge_year'][mid])
  merge_month_np = np.array(f.variables['merge_month'][mid])
  merge_day_np   = np.array(f.variables['merge_day'][mid])
  merge_hour_np  = np.array(f.variables['merge_hour'][mid])
  merge_min_np   = np.array(f.variables['merge_min'][mid])
  valid_time     = dt.datetime(merge_year_np,merge_month_np,merge_day_np,merge_hour_np,merge_
↪min_np,0)
  return valid_time
```

(continues on next page)

(continued from previous page)

```
def read_inputs():
    # Read the input file as the first argument
    input_file   = os.path.expandvars(sys.argv[1])
    var_name     = sys.argv[2]
    mission_name = sys.argv[3]
    level_km     = float(sys.argv[4])
    return input_file, var_name, mission_name, level_km

def main(input_file, var_name, mission_name, level_km):
  ########################################

  ##
  ##  input file specified on the command line
  ##  load the data into the numpy array
  ##


    try:
      # Print some output to verify that this script ran
      print("Input File:      " + repr(input_file))
      print("Variable Name:   " + repr(var_name))

      # Read input file
      f = Dataset(input_file, 'r')

      # Find the requested mission name
      mid = f.variables['mission_ID'][:].tolist().index(mission_name)

      # Find the requested level value
      lid = f.variables['level'][:].tolist().index(level_km)

      # Read the requested variable
      data = np.float64(f.variables[var_name][mid,:,:,lid])

      # Expect that dimensions are ordered (lat, lon)
      # If (lon, lat), transpose the data
      if(f.variables[var_name].dimensions[0] == 'lon'):
         data = data.transpose()

      print("Mission (index): " + repr(mission_name) + " (" + repr(mid) + ")")
      print("Level (index):   " + repr(level_km) + " (" + repr(lid) + ")")
      print("Data Range:      " + repr(np.nanmin(data)) + " to " + repr(np.nanmax(data)))

      # Reset any negative values to missing data (-9999 in MET)
      data[np.isnan(data)] = -9999
```

(continues on next page)

(continued from previous page)

```
    # Flip data along the equator
    data = data[::-1]

    # Store a deep copy of the data for MET
    met_data = data.reshape(200,200).copy()

    print("Data Shape:      " + repr(met_data.shape))
    print("Data Type:       " + repr(met_data.dtype))

  except NameError:
    print("Trouble reading input file: " . input_file)


  ################################################################################

  # Determine LatLon grid information

  # Read in coordinate data
  merged_lon  = np.array(f.variables['merged_longitudes'][mid,0,:])
  merged_lat  = np.array(f.variables['merged_latitudes'][mid,:,0])

  # Time data:
  valid_time = calculate_valid_time(f, mid)
  init_time = valid_time

  ##########################################

  ##
  ##  create the metadata dictionary
  ##

  ##########################################
  attrs = {
    'valid': valid_time.strftime("%Y%m%d_%H%M%S"),
    'init' : valid_time.strftime("%Y%m%d_%H%M%S"),
    'lead':  '00',
    'accum': '06',
    'mission_id': mission_name,

    'name':      var_name,
    'long_name': var_name,
    'level':     str(level_km) + "km",
    'units':     str(getattr(f.variables[var_name], "units")),
```

(continues on next page)

(continued from previous page)

```
      'grid': {
          'name':       var_name,
          'type' :      'LatLon',
          'lat_ll' :    float(min(merged_lat)),
          'lon_ll' :    float(min(merged_lon)),
          'delta_lat' : float(merged_lat[1]-merged_lat[0]),
          'delta_lon' : float(merged_lon[1]-merged_lon[0]),
          'Nlat' :      len(merged_lat),
          'Nlon' :      len(merged_lon),
      }
    }

    print("Attributes:      " + repr(attrs))
    return met_data, attrs

if __name__ == '__main__':
    if len(sys.argv) < 5:
        print("Must specify exactly one input file, variable name, mission ID (YYMMDDID),␣
↪level (in km)")
        sys.exit(1)

    input_file, var_name, mission_name, level_km = read_inputs()

    met_data, attrs = main(input_file, var_name, mission_name, level_km)
```

Running METplus

This use case can be run two ways:

1) Passing in GridStat_fcstHAFS_obsTDR_NetCDF.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications//tc_and_extra_tc/
↪GridStat_fcstHAFS_obsTDR_NetCDF.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in GridStat_fcstHAFS_obsTDR_NetCDF.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/tc_and_extra_tc/
↪GridStat_fcstHAFS_obsTDR_NetCDF.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases

- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in nam (relative to **OUTPUT_BASE**) and will contain the following files:

- grid_stat_HAFS_vs_TDR_000000L_20190829_120000V_fho.txt
- grid_stat_HAFS_vs_TDR_000000L_20190829_120000V_pairs.nc
- grid_stat_HAFS_vs_TDR_000000L_20190829_120000V.stat
- The use case is run for 4 lead times valid at 2019081912, so four directories will be generated which contains similar files as above.

Keywords

Note:

- TCandExtraTCAppUseCase
- GridStatToolUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/tc_and_extra_tc-GridStat_fcstHAFS_obsTDR_NetCDF.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.15.4 CyclonePlotter: Extra-TC Tracker and Plotting Capabilities

model_applications/tc_and_extra_tc/CyclonePlotter_fcstGFS_obsGFS_UserScript_ExtraTC.conf

Scientific Objective

Once this method is complete, a user-created extra TC track file for the valid date of interest (YYYYMMDDHH) will have been created, paired up by TCPairs, and global storm tracks for the valid date of interest will be plotted by CyclonePlotter (PlateCaree projection)

Datasets

Forecast: Adeck

/path/to/{init?fmt=%Y}/trak.gfso.atcf_gen.glbl.{init?fmt=%Y}

Observation: Bdeck

/path/to/{init?fmt=%Y}/trak.gfso.atcf_gen.glbl.{init?fmt=%Y}

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1930) section for more information.

Data Source: GFS

External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- cartopy
- matplotlib

METplus Components

This use case utilizes Python user script-created output files that are accessible via the TCPairs wrapper. Due to the nature of the source file (already tracked extra TCs), the TCPairs wrapper is passed the "Adeck" file for each storm twice: once as the adeck or forecast file, and once as the bdeck or analysis file. Essentially, TCPairs is matching a forecast to itself. It then uses the CyclonePlotter wrapper to create a global plot of storm tracks for the desired day of interest (YYYYMMDDHH).

METplus Workflow

TCPairs is the first tool called in this example. It processes the following run times for each storm file:

Init/Valid: 2020100700

CyclonePlotter is the second (and final) tool called in this example. It processes the output from TCPairs.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c /path/to/TCPairs_extra_tropical.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/tc_and_extra_tc/
↪CyclonePlotter_fcstGFS_obsGFS_UserScript_ExtraTC.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript, TCPairs, CyclonePlotter


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
```

(continues on next page)

(continued from previous page)

```
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2020100700
INIT_END = 2020100700
INIT_INCREMENT = 21600

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

USER_SCRIPT_OUTPUT_DIR = {OUTPUT_BASE}/decks

TC_PAIRS_ADECK_INPUT_DIR = {USER_SCRIPT_OUTPUT_DIR}/adeck
TC_PAIRS_ADECK_TEMPLATE = adeck.{init?fmt=%Y%m%d%H}.{cyclone}.dat

TC_PAIRS_BDECK_INPUT_DIR = {USER_SCRIPT_OUTPUT_DIR}/adeck
TC_PAIRS_BDECK_TEMPLATE = adeck.{init?fmt=%Y%m%d%H}.{cyclone}.dat

TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs
TC_PAIRS_OUTPUT_TEMPLATE = tc_pairs.{init?fmt=%Y%m%d%H}.{cyclone}

CYCLONE_PLOTTER_INPUT_DIR = {TC_PAIRS_OUTPUT_DIR}
CYCLONE_PLOTTER_OUTPUT_DIR = {OUTPUT_BASE}/cyclone


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

USER_SCRIPT_PATH = {PARM_BASE}/use_cases/model_applications/tc_and_extra_tc/CyclonePlotter_
```

(continues on next page)

(continued from previous page)

```
↪fcstGFS_obsGFS_UserScript_ExtraTC/extract_opc_decks.py

USER_SCRIPT_INPUT_PATH = {INPUT_BASE}/model_applications/tc_and_extra_tc/CyclonePlotter_
↪fcstGFS_obsGFS_UserScript_ExtraTC/trak.gfso.atcf_gen.glbl.{init?fmt=%Y}

USER_SCRIPT_COMMAND = {USER_SCRIPT_PATH} {USER_SCRIPT_INPUT_PATH} {USER_SCRIPT_OUTPUT_DIR}
↪{init?fmt=%Y%m%d%H}


###
# TCPairs Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcpairs
###

TC_PAIRS_DLAND_FILE = MET_BASE/tc_data/dland_global_tenth_degree.nc

TC_PAIRS_MATCH_POINTS = FALSE


###
# CyclonePlotter Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#cycloneplotter
###

CYCLONE_PLOTTER_INIT_DATE={init?fmt=%Y%m%d}
CYCLONE_PLOTTER_INIT_HR ={init?fmt=%H}
CYCLONE_PLOTTER_MODEL = GFSO
CYCLONE_PLOTTER_PLOT_TITLE = Model Forecast Storm Tracks

CYCLONE_PLOTTER_GLOBAL_PLOT = no

CYCLONE_PLOTTER_WEST_LON = -180
CYCLONE_PLOTTER_EAST_LON = 179
CYCLONE_PLOTTER_SOUTH_LAT = 0
CYCLONE_PLOTTER_NORTH_LAT = 90

CYCLONE_PLOTTER_CIRCLE_MARKER_SIZE = 4
CYCLONE_PLOTTER_CROSS_MARKER_SIZE = 6

CYCLONE_PLOTTER_ANNOTATION_FONT_SIZE = 3

CYCLONE_PLOTTER_LEGEND_FONT_SIZE = 3

CYCLONE_PLOTTER_RESOLUTION_DPI = 400
```

(continues on next page)

(continued from previous page)

```
CYCLONE_PLOTTER_GENERATE_TRACK_ASCII = yes

CYCLONE_PLOTTER_ADD_WATERMARK = False
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TCPairs MET Configuration* (page 282) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

//
// Models
//
${METPLUS_MODEL}

//
// Description
//
${METPLUS_DESC}

//
// Storm identifiers
//
${METPLUS_STORM_ID}
```

(continues on next page)

(continued from previous page)

```
//
// Basins
//
${METPLUS_BASIN}

//
// Cyclone numbers
//
${METPLUS_CYCLONE}

//
// Storm names
//
${METPLUS_STORM_NAME}

//
// Model initialization time windows to include or exclude
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
${METPLUS_VALID_INC}
// valid_exc =
${METPLUS_VALID_EXC}

// write_valid =
${METPLUS_WRITE_VALID}

//
// Valid model time window
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}

//
// Model initialization hours
//
init_hour = [];

//
```

(continues on next page)

(continued from previous page)

```
// Required lead time in hours
//
lead_req = [];

//
// lat/lon polylines defining masking regions
//
init_mask  = "";
valid_mask = "";

//
// Specify if the code should check for duplicate ATCF lines
//
//check_dup =
${METPLUS_CHECK_DUP}


//
// Specify special processing to be performed for interpolated models.
// Set to NONE, FILL, or REPLACE.
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
//
//consensus =
${METPLUS_CONSENSUS_LIST}


//
// Forecast lag times
//
lag_time = [];

//
// CLIPER/SHIFOR baseline forecasts to be derived from the BEST
// and operational (CARQ) tracks.
//
best_technique = [ "BEST" ];
best_baseline  = [];
oper_technique = [ "CARQ" ];
oper_baseline  = [];
```

(continues on next page)

(continued from previous page)

```
//
// Specify the datasets to be searched for analysis tracks (NONE, ADECK, BDECK,
// or BOTH).
//
anly_track = BDECK;

//
// Specify if only those track points common to both the ADECK and BDECK
// tracks be written out.
//
//match_points =
${METPLUS_MATCH_POINTS}

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
${METPLUS_DLAND_FILE}

//
// Specify watch/warning information:
//   - Input watch/warning filename
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}


//diag_info_map = {
${METPLUS_DIAG_INFO_MAP_LIST}

//diag_convert_map = {
${METPLUS_DIAG_CONVERT_MAP_LIST}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses a Python embedding script to read input data. Because the source file already contains "analysis" tracks for the extra TCs, this Python script only needs to output storm tracks that have a valid time matching the user input. These storms are put into separate storm files, to better mimic how TC storms are typically passed to TCPairs.

parm/use_cases/model_applications/tc_and_extra_tc/CyclonePlotter_fcstGFS_obsGFS_UserScript_ExtraTC/extract_op

```
#! /usr/bin/env python3

#
#  program extrack_opc_decks.py
#
#  reads in EMC 2020 cyclone data
#  takes 3 command line arguments
#  1) input file (full path, eg, "/d2/projects/d2/projects/extra-tc_verif/gpfs/dell1/nco/ops/
↪com/gentracks/prod/gentracks/{init?fmt=%Y}/trak.gfso.atcf_gen.glbl.{init?fmt=%Y}"
#  2) output directory (eg "{OUTPUT_BASE}/decks")
#  3) init time (YYYYMMDDHH)
#
#  reads all data in input file, creates ADECK using all points valid at init time (key
↪'YYYYMMDDHH', creates BDECK
#    using key ('STORMNAME') for all storms in ADECK where forecast key ('TAU') = '000' or 0␣
↪hrs
#  writes a single adeck and a single bdeck file containing all storms
#
#  further processed by TC_Pairs (extra-tropical) and CyclonePlotter in single use-case␣
↪wrapper CyclonePlotter_fcst_GFS_obsGFS_OPC
#
#  written February 2021 by George McCabe (mccabe@ucar.edu)
#

import sys
import os
import pandas as pd

# column names/dictionary keys for the trak.data file
atcf_headers_trak=['BASIN','CYCLONE','STORMNAME','YYYYMMDDHH','TECHNUM/MIN','TECH','TAU','LAT
↪','LON',
                   'VMAX','MSLP','TY','RAD','WINDCODE','RAD1','RAD2','RAD3','RAD4','POUTER',
                   'ROUTER','RMW','GUSTS','EYE','SUBREGION','MAXSEAS','INITIALS','DIR','SPEED
↪','F1','F2',
                   'STORMNAME2','DEPTH','SEAS','SEASCODE','SEAS1','SEAS2','SEAS3','SEAS4']

# needs exactly 3 arguments (see above)
num_args = len(sys.argv) - 1
```

(continues on next page)

(continued from previous page)

```
if num_args < 3:
    print("ERROR: Not enough arguments")
    sys.exit(1)
debug = 'debug' in sys.argv
# function to compare storm warning time to search time
def is_equal(column_val, search_string):
    return str(column_val).strip() == search_string

input_file = sys.argv[1]
output_dir = sys.argv[2]
search_date = sys.argv[3]

if debug:
    print(f"Running {__file__}\nSearch date: {search_date}")

# get 2 digit year to use in CYCLONE column substitute value
search_year = search_date[2:4]

# string to use in output file names for filtered adeck and bdeck files
file_prefix = f'deck.{search_date}.'

# an intermediate directory path for the separate files
adeck_base = os.path.join(output_dir, "adeck")
#bdeck_base = os.path.join(output_dir, "bdeck")

# create output directories if not already there
if not os.path.exists(adeck_base):
    print(f"Creating output directory: {adeck_base}")
    os.makedirs(adeck_base)

#if not os.path.exists(bdeck_base):
#    print(f"Creating output directory: {bdeck_base}")
#    os.makedirs(bdeck_base)

# using pandas (pd), read input file
print(f"Reading input file: {input_file}")
pd_data = pd.read_csv(input_file, names=atcf_headers_trak)

print(f"Filtering data...")

# get all 0 hour analyses data
print(f"Filtering data 0 (hr) in TAU (forecast hour) column for bdeck")
pd_0hr_data = pd_data[pd_data['TAU'] == 0]
```

(continues on next page)

(continued from previous page)

```
# get adeck - all lines that match the desired date for YYYYMMDDHH (init time)
print(f"Filtering data with {search_date} in YYYYMMDDHH column for adeck")
init_matches = pd_data['YYYYMMDDHH'].apply(is_equal,
                                           args=(search_date,))
adeck = pd_data[init_matches]

# get list of STORMNAMEs from adeck data
all_storms = adeck.STORMNAME.unique()

# initialize counter to use to set output filenames with "cyclone" number
# to keep storms in separate files
index = 0

# loop over storms
for storm_name in all_storms:
    index_pad = str(index).zfill(4)

    # remove whitespace at beginning of storm name
    storm_name = storm_name.strip()

    # get 0hr data for given storm to use as bdeck
    storm_b_match = pd_0hr_data['STORMNAME'].apply(is_equal,
                                                   args=(storm_name,))
    storm_bdeck = pd_0hr_data[storm_b_match]
    if debug:
        print(f"Processing storm: {storm_name}")
    wrote_a = wrote_b = False

    #Logic for writing out Analysis files. Currently commented out,
    #but left in for possible future use
    if not storm_bdeck.empty:
    #    bdeck_filename = f'b{file_prefix}{index_pad}.dat'
    #    bdeck_path = os.path.join(bdeck_base, bdeck_filename)

    #    print(f"Writing bdeck to {bdeck_path}")
    #    storm_bdeck.to_csv(bdeck_path, header=False, index=False)
        wrote_b = True
    #else:
    #    print(f"BDECK for {storm_name} is empty. Skipping")

    # filter out adeck data for given storm
    storm_a_match = adeck['STORMNAME'].apply(is_equal,
                                             args=(storm_name,))
    storm_adeck = adeck[storm_a_match]
```

(continues on next page)

(continued from previous page)

```
    if not storm_adeck.empty:
        adeck_filename = f'a{file_prefix}{index_pad}.dat'
        adeck_path = os.path.join(adeck_base, adeck_filename)
        if debug:
            print(f"Writing adeck to {adeck_path}")
        storm_adeck.to_csv(adeck_path, header=False, index=False)
        wrote_a = True
    else:
        if debug:
            print(f"ADECK for {storm_name} is empty. Skipping")

    if wrote_a or wrote_b:
        index += 1

print("Finished processing all storms")
```

Running METplus

It is recommended to run this use case by:

Passing in TCPairs_extra_tropical.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/CyclonePlotter_fcstGFS_obsGFS_UserScript_ExtraTC.conf -c /path/to/
↪user_system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where EMC data files (csv) are read (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in **tc_pairs/201412** (relative to **OUTPUT_BASE**) and will contain the following files:

- decks/adeck/adeck.2020100700.xxxx.dat
- tc_pairs/tc_pairs.2020100700.xxxx.tcst
- cyclone/20201007.png
- cyclone/20201007.txt

where "xxxx" is the unique four digit storm identifier for TCPairs wrapper to use.

Keywords

Note:

- TCPairsToolUseCase
- SBUOrgUseCase
- CyclonePlotterUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/tc_and_extra_tc-CyclonePlotter_fcstGFS_obsGFS_UserScript_ExtraTC.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.15.5 Point-Stat: Standard Verification for CONUS Surface

model_applications/tc_and_extra_tc/UserScript_ASCII2NC_PointStat_fcstHAFS_obsFRD _NetCDF.conf

Scientific Objective

To provide useful statistical information on the relationship between observation data in point format to a gridded forecast. These values can be used to assess the skill of the prediction. Statistics are store as partial sums to save space and Stat-Analysis must be used to compute Continuous statistics.

Datasets

Forecast: HAFS temperature
Observation: HRD Dropsonde data

Location of Model forecast and Dropsonde files: All of the input data required for this use case can be found in the sample data tarball. Click here to download.
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1946) section for more information.

Dropsonde Data Source: Hurricane Research Division Sonde Archive

METplus Components

This use case utilizes the METplus ASCII2NC wrapper to convert full-resolution data (frd) dopsonde point observations to NetCDF format and then compare them to gridded forecast data using PointStat.

METplus Workflow

The use case runs the UserScript wrapper (untar the dropsonde file and extract the files to a directory), ASCII2NC (convert the ascii files to NetCDF format), and PointStat (compute statistics against HAFS model output), which are the tools called in this example. It processes the following run times:

Valid: 2019-08-29 12Z

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/tc_and_extra_tc/UserScript_ASCII2NC_PointStat_fcstHAFS_obsFRD_NetCDF.con

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/tc_and_extra_tc/
↪UserScript_ASCII2NC_PointStat_fcstHAFS_obsFRD_NetCDF.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = UserScript(untar_drop_file), Ascii2nc, PointStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2019082912
VALID_END = 2019082912
VALID_INCREMENT = 21600

LEAD_SEQ = 0,6,12,18

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID


###
```

(continues on next page)

(continued from previous page)

```
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

# UserScript

USER_SCRIPT_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/dropsonde/obs
USER_SCRIPT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/tc_and_extra_tc/dropsonde/obs


# ASCII2NC

ASCII2NC_INPUT_TEMPLATE = "{PARM_BASE}/use_cases/model_applications/tc_and_extra_tc/
↪UserScript_ASCII2NC_PointStat_fcstHAFS_obsFRD_NetCDF/hrd_frd_sonde_for_ascii2nc.py {USER_
↪SCRIPT_OUTPUT_DIR}/{valid?fmt=%Y%m%d}"

ASCII2NC_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/tc_and_extra_tc/dropsonde/ascii2nc
ASCII2NC_OUTPUT_TEMPLATE = drop{valid?fmt=%Y%m%d}.nc

# PointStat

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/dropsonde
FCST_POINT_STAT_INPUT_TEMPLATE = hafs.{valid?fmt=%Y%m%d%H}/dorian05l.{init?fmt=%Y%m%d%H}.
↪hafsprs.synoptic.TMP600-900.0p03.f{lead?fmt=%3H}.grb2

OBS_POINT_STAT_INPUT_DIR = {OUTPUT_BASE}/model_applications/tc_and_extra_tc/dropsonde/
↪ascii2nc
OBS_POINT_STAT_INPUT_TEMPLATE = {ASCII2NC_OUTPUT_TEMPLATE}

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/{OBTYPE}


###
# Field Info
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#field-info
###

MODEL = HAFS
OBTYPE = drop

BOTH_VAR1_NAME = TMP
BOTH_VAR1_LEVELS = P925-950, P850-800, P700-650

OBS_WINDOW_BEGIN = -5400
```

(continues on next page)

(continued from previous page)

```
OBS_WINDOW_END = 5400


###
# UserScript Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#userscript
###

USER_SCRIPT_ARGUMENTS = {USER_SCRIPT_INPUT_DIR} {valid?fmt=%Y%m%d} {USER_SCRIPT_OUTPUT_DIR}
USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/model_applications/tc_and_extra_tc/UserScript_
↪ASCII2NC_PointStat_fcstHAFS_obsFRD_NetCDF/hrd_frd_sonde_find_tar.py {USER_SCRIPT_ARGUMENTS}


###
# ASCII2NC Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#ascii2nc
###

ASCII2NC_INPUT_FORMAT = python
ASCII2NC_TIME_SUMMARY_FLAG = False
ASCII2NC_TIME_SUMMARY_RAW_DATA = False
ASCII2NC_TIME_SUMMARY_BEG = 000000
ASCII2NC_TIME_SUMMARY_END = 235959
ASCII2NC_TIME_SUMMARY_STEP = 300
ASCII2NC_TIME_SUMMARY_WIDTH = 600
ASCII2NC_TIME_SUMMARY_GRIB_CODES = 11, 204, 211
ASCII2NC_TIME_SUMMARY_VAR_NAMES =
ASCII2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80
ASCII2NC_TIME_SUMMARY_VALID_FREQ = 0
ASCII2NC_TIME_SUMMARY_VALID_THRESH = 0.0


###
# PointStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#pointstat
###

POINT_STAT_MESSAGE_TYPE = ADPUPA

POINT_STAT_GRID = FULL

POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD = NEAREST

POINT_STAT_INTERP_TYPE_METHOD = BILIN
POINT_STAT_INTERP_TYPE_WIDTH = 2
```

(continues on next page)

(continued from previous page)

```
POINT_STAT_OUTPUT_FLAG_SL1L2 = STAT
POINT_STAT_OUTPUT_FLAG_VL1L2 = STAT
POINT_STAT_OUTPUT_FLAG_FHO = BOTH
POINT_STAT_OUTPUT_FLAG_CTC = BOTH
POINT_STAT_OUTPUT_FLAG_CTS = STAT
POINT_STAT_OUTPUT_FLAG_CNT = BOTH
POINT_STAT_OUTPUT_FLAG_ECLV = BOTH
POINT_STAT_OUTPUT_FLAG_MPR = BOTH

POINT_STAT_REGRID_TO_GRID = NONE
```

Notes for USER_SCRIPT* METplus conf items for this use case:

- **${USER_SCRIPT_RUNTIME_FREQ}** - Corresponds to USER_SCRIPT_RUNTIME_FREQ in the METplus configuration file.
- **${USER_SCRIPT_INPUT_DIR}** - Corresponds to USER_SCRIPT_INPUT_DIR in the METplus configuration file.
- **${USER_SCRIPT_OUTPUT_DIR}** - Corresponds to USER_SCRIPT_OUTPUT_DIR in the METplus configuration file.
- **${USER_SCRIPT_COMMAND}** - Arguments needed to hrd_frd_sonde_find_tar.py corresponds to USER_SCRIPT_INPUT_TEMPLATE.
- **${USER_SCRIPT_INPUT_TEMPLATE}** - Input template to hrd_frd_sonde_find_tar.py: USER_SCRIPT_INPUT_DIR, valid date (%Y%m%d), and USER_SCRIPT_OUTPUT_DIR.

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Ascii2NcConfig_wrapped

Note: See the *ASCII2NC MET Configuration* (page 98) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
```

(continues on next page)

(continued from previous page)

```
// Default ascii2nc configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to summarize the ASCII data read in
//

//
// Time periods for the summarization
// obs_var (string array) is added and works like grib_code (int array)
// when the obs name is given instead of grib_code
//
${METPLUS_TIME_SUMMARY_DICT}

//
// Mapping of input little_r report types to output message types
//
message_type_map = [
   { key = "FM-12 SYNOP";  val = "ADPSFC"; },
   { key = "FM-13 SHIP";   val = "SFCSHP"; },
   { key = "FM-15 METAR";  val = "ADPSFC"; },
   { key = "FM-18 BUOY";   val = "SFCSHP"; },
   { key = "FM-281 QSCAT"; val = "ASCATW"; },
   { key = "FM-32 PILOT";  val = "ADPUPA"; },
   { key = "FM-35 TEMP";   val = "ADPUPA"; },
   { key = "FM-88 SATOB";  val = "SATWND"; },
   { key = "FM-97 ACARS";  val = "AIRCFT"; }
];

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

PointStatConfig_wrapped

Note: See the *PointStat MET Configuration* (page 218) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Point-Stat configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Output model name to be written
//
// model =
${METPLUS_MODEL}

//
// Output description to be written
// May be set separately in each "obs.field" entry
//
// desc =
${METPLUS_DESC}

////////////////////////////////////////////////////////////////////////////////

//
// Verification grid
//
// regrid = {
${METPLUS_REGRID_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// May be set separately in each "field" entry
//
censor_thresh = [];
censor_val    = [];
cat_thresh    = [ NA ];
cnt_thresh    = [ NA ];
cnt_logic     = UNION;
wind_thresh   = [ NA ];
wind_logic    = UNION;
eclv_points   = 0.05;
//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;
```

(continues on next page)

(continued from previous page)

```
//
// Forecast and observation fields to be verified
//
fcst = {
  ${METPLUS_FCST_FILE_TYPE}
  ${METPLUS_FCST_FIELD}
}

obs = {
  ${METPLUS_OBS_FILE_TYPE}
  ${METPLUS_OBS_FIELD}
}
////////////////////////////////////////////////////////////////////////////////

//
// Point observation filtering options
// May be set separately in each "obs.field" entry
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}

duplicate_flag = NONE;
obs_summary    = NONE;
obs_perc_value = 50;

//
// Mapping of message type group name to comma-separated list of values.
//
//message_type_group_map =
${METPLUS_MESSAGE_TYPE_GROUP_MAP}

////////////////////////////////////////////////////////////////////////////////

//
// Climatology data
//
//climo_mean = {
${METPLUS_CLIMO_MEAN_DICT}
```

(continues on next page)

(continued from previous page)

```
//climo_stdev = {
${METPLUS_CLIMO_STDEV_DICT}

//
// May be set separately in each "obs.field" entry
//
//climo_cdf = {
${METPLUS_CLIMO_CDF_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Point observation time window
//
// obs_window = {
${METPLUS_OBS_WINDOW_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Verification masking regions
//
//mask = {
${METPLUS_MASK_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

////////////////////////////////////////////////////////////////////////////////

//
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

////////////////////////////////////////////////////////////////////////////////
// Threshold for SEEPS p1 (Probability of being dry)

//seeps_p1_thresh =
${METPLUS_SEEPS_P1_THRESH}

////////////////////////////////////////////////////////////////////////////////

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

////////////////////////////////////////////////////////////////////////////////

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses two Python embedding scripts: one to download the data (hrd_frd_sonde_find_tar.py) and the other to process it (hrd_frd_sonde_for_ascii2nc.py).

parm/use_cases/model_applications/tc_and_extra_tc/UserScript_ASCII2NC_PointStat_fcstHAFS_obsFRD_NetCDF/hr

```
#! /usr/bin/env python3
########################################################################
#  This script will untar the FRD formatted dropsonde tar files from
#  https://www.aoml.noaa.gov/hrd/data_sub/dropsonde.htmli
#  The untarred files will be downloaded in to a direcory
#  under USER_SCRIPT_OUTPUT_DIR. Arguments to the scripts includes
#  directory where the tar files exists, the user specified
#  date in YYYYMMDD, and output directory
#  Author: biswas@ucar.edu
########################################################################

import sys
import os
import glob
import tarfile

if len(sys.argv) == 4:
  path = sys.argv[1]
  date = sys.argv[2]
  outdir = sys.argv[3]

  if os.path.exists(path):
     print("Directory exists: "+ path)

     for name in glob.glob(path+'/'+str(date)+'*FRD.tar.gz'):
       print (name)

       drop_tar = tarfile.open(name)
       drop_tar.extractall(outdir + '/'+str(date))
       drop_files = os.listdir(outdir + '/'+str(date))
       print(drop_files)
       drop_tar.close()

  else:
     print("Directory not present" + path)

else:
  print("ERROR : Must specify exactly one input data directory, date (YYYYMMDD), and output␣
↪directory.")
  sys.exit(1)
```

(continues on next page)

(continued from previous page)

```
######################################################################
```

parm/use_cases/model_applications/tc_and_extra_tc/UserScript_ASCII2NC_PointStat_fcstHAFS_obsFRD_NetCDF/hrd

```
######################################################################
#
# Description:
#   Prepare HRD FRD (full-resolution data) dropsonde files for further
#   processing by the ascii2nc tool in MET.
#   Source: https://www.aoml.noaa.gov/hrd/data_sub/dropsonde.html
#
# Date:
#   December 2020
#
######################################################################

import re
import os
import sys
import numpy as np
import itertools
import datetime as dt
from datetime import datetime, timedelta
import pandas as pd

# Check arguments
if len(sys.argv) == 2:
  input_dir = os.path.expandvars(sys.argv[1])
  print("Input Dir:\t" + repr(input_dir))
else:
  print("ERROR:", sys.argv[0],
        "-> Must specify exactly one input file.")
  sys.exit(1)

# Empty object
my_data = pd.DataFrame()

for filename in sorted(os.listdir(input_dir)):
   input_file = os.path.join(input_dir, filename)

   # Open file
   with open(input_file, 'r') as file_handle:
      lines = file_handle.read().splitlines()
   readdata = False
   for idx, line in enumerate(lines):
```

(continues on next page)

(continued from previous page)

```
# Extract date, time and sonde info
   match_date = re.match(r'^ Date:(.*)', line)
   match_time = re.match(r'^ Time:(.*)', line)
   match_sonde = re.match(r'^ SID:(.*)', line)

   if match_date:
     date_items = match_date.group(1).split()[:1]
     lat = match_date.group(1).split()[:4]
   if match_time:
     time_items = match_time.group(1).split()[:1]
     lon = match_time.group(1).split()[:4]
   if match_sonde:
     sonde = match_sonde.group(1).split()[0]

     # Format the date and time
     date_formatted = \
       f"{date_items[0][:2]}{date_items[0][2:4]}{date_items[0][4:6]}_" +\
       f"{time_items[0][:2]}:{time_items[0][2:4]}:{time_items[0][4:6]}"
     valid_time = \
       dt.datetime.strptime(date_formatted, "%y%m%d_%H:%M:%S")
     print(f"Valid Time:\t{valid_time}")
   if line.startswith("IX"):
       readdata = True
       continue
   if not readdata:
       continue
   line    = line.strip()
   columns = line.split()
   dsec    = str(columns[1])     # time elasp (s)
   pres    = float(columns[2])   # pressure (mb)
   temp    = float(columns[3])   # temperature (C)
   temp    = temp + 273.15        # convert deg C to K
   relh    = float(columns[4])   # relative humidity (%)
   geop    = int(columns[5])     # geopotential mass height (m)
   wind_dir = int(columns[6])    # wind direction (E)
   wind_spd = float(columns[7])  # wind speed (m/s)
   wind_z   = float(columns[8])  # zonal wind (m/s)
   wind_m   = float(columns[9])  # meridional wind (m/s)
   wind_w   = float(columns[11]) # vertical velocity (m/s)
   zw       = int(columns[12])   # geopotential wind height (m)
   lat      = float(columns[17]) # lat (N)
   lon      = float(columns[18]) # lon (E)
   vld      = valid_time + dt.timedelta(seconds=float(dsec))
```

(continues on next page)

(continued from previous page)

```
      # Skip line if dsec, lat, or lon are missing.
      # Or if pres and geop are missing.
      if dsec == -999.0 or lat == -999.0 or lon == -999.0 or +\
          (pres == -999.0 and geop == -999):
         continue

      # Store valid time in YYYYMMDD_HHMMSS format
      t_vld = vld.strftime('%Y%m%d_%H%M%S')

      # Flag values for the station elevation and qc
      elv = "-9999"
      qc  = "-9999"

      # Append observations for this line
      # Name variable using GRIB conventions:
      #   https://www.nco.ncep.noaa.gov/pmb/docs/on388/table2.html
      if temp != -999.0:
         my_data = my_data.append(pd.DataFrame(np.array(
           ["ADPUPA", str(sonde), t_vld, lat, lon, elv, \
             "TMP", pres, geop, qc, temp]])))

      if relh != -999.0:
         my_data = my_data.append(pd.DataFrame(np.array(
           ["ADPUPA", str(sonde), t_vld, lat, lon, elv, \
             "RH", pres, geop, qc, relh]])))

      if geop != -999.0 and pres != -999.0:
         my_data = my_data.append(pd.DataFrame(np.array(
           ["ADPUPA", str(sonde), t_vld, lat, lon, elv, \
             "HGT", pres, geop, qc, geop]])))

      if wind_dir != -999.0:
         my_data = my_data.append(pd.DataFrame(np.array(
           ["ADPUPA", str(sonde), t_vld, lat, lon, elv, \
             "WDIR", pres, zw, qc, wind_dir]])))

      if wind_spd != -999.0:
         my_data = my_data.append(pd.DataFrame(np.array(
           ["ADPUPA", str(sonde), t_vld, lat, lon, elv, \
             "WIND", pres, zw, qc, wind_spd]])))

      if wind_z != -999.0:
         my_data = my_data.append(pd.DataFrame(np.array(
           ["ADPUPA", str(sonde), t_vld, lat, lon, elv, \
             "UGRD", pres, zw, qc, wind_z]])))
```

(continues on next page)

(continued from previous page)

```
      if wind_m != -999.0:
         my_data = my_data.append(pd.DataFrame(np.array(
           [["ADPUPA", str(sonde), t_vld, lat, lon, elv, \
             "VGRD", pres, zw, qc, wind_m]])))

      if wind_w != -999.0:
         my_data = my_data.append(pd.DataFrame(np.array(
           [["ADPUPA", str(sonde), t_vld, lat, lon, elv, \
             "DZDT", pres, zw, qc, wind_w]])))

# Prepare point_data object for ascii2nc
point_data = my_data.values.tolist()
print("Data Length:\t" + repr(len(point_data)))
print("Data Type:\t" + repr(type(point_data)))
```

Running METplus

This use case can be run two ways:

1) Passing in UserScript_ASCII2NC_PointStat_fcstHAFS_obsFRD_NetCDF.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications//tc_and_extra_tc/
↪UserScript_ASCII2NC_PointStat_fcstHAFS_obsFRD_NetCDF.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_ASCII2NC_PointStat_fcstHAFS_obsFRD_NetCDF.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/tc_and_extra_tc/UserScript_ASC
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
```

(continues on next page)

(continued from previous page)

```
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in nam (relative to **OUTPUT_BASE**) and will contain the following files:

- point_stat_180000L_20190829_120000V.stat
- point_stat_180000L_20190829_120000V_fho.txt
- point_stat_180000L_20190829_120000V_eclv.txt
- point_stat_180000L_20190829_120000V_ctc.txt
- point_stat_180000L_20190829_120000V_cnt.txt
- point_stat_180000L_20190829_120000V_mpr.txt

Keywords

Note:

- TCandExtraTCAppUseCase
- UserScriptUseCase
- PointStatToolUseCase
- ASCII2NCToolUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/tc_and_extra_tc-UserScript_ASCII2NC_PointStat_fcstHAFS_obsFRD_NetCDI

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.15.6 TCGen: 2021 Global Forecast System (GFS) Tropical Cyclone Genesis Forecast

model_applications/tc_and_extra_tc/TCGen_fcstGFS_obsBDECK_2021season.conf

Scientific Objective

This use case runs TC-Gen to analyze the operational Global Forecast System (GFS) tropical cyclone (TC) genesis forecasts for a portion of the 2021 Atlantic and Eastern Pacific basin hurrican seasons. TC-Gen will produce verification of deterministic and probabilistic tropical cyclone genesis forecasts in the ATCF file and shape file formats. TC-Gen will output deterministic and probabilistic categorical counts and statistics and genesis matched pairs, which is a specific line type for TC-Gen.

Datasets

Forecast: GFS genesis file, GFS E Deck
Observation: B Deck, A Deck
Warning Areas: Shapefiles

Location: All of the input data required for this use case can be found in the tc_and_extra_tc sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
This tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1959) section for more information.

Data Source: NHC ftp.noaa.gov/atcf
Data Source: www.nhc.noaa.gov/archive/wgtwo/

METplus Components

This case utilizes the METplus TC-Gen wrapper to run TC-Gen for deterministic and probabilistic genesis forecasts with ASCII and netcdf output.

METplus Workflow

TC-Gen is the only tool called in this example. It processes the following run times:

Init: 2021-05-07 00 UTC - 2021-11-13 12 UTC
Forecast lead: 06 - 120 hours

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/tc_and_extra_tc/TCGen_fcstGFS_obsBDECK_2021season.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/tc_and_extra_tc/
↪TCGen_fcstGFS_obsBDECK_2021season.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCGen


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
```

(continues on next page)

(continued from previous page)

```
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y
INIT_BEG = 2021


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

TC_GEN_TRACK_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/TCGen_fcstGFS_
↪obsBDECK_2021season/abdeck/
TC_GEN_TRACK_INPUT_TEMPLATE = *.dat

TC_GEN_GENESIS_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/TCGen_fcstGFS_
↪obsBDECK_2021season/genesis/
TC_GEN_GENESIS_INPUT_TEMPLATE = genesis*.atcf_gen

TC_GEN_EDECK_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/TCGen_fcstGFS_
↪obsBDECK_2021season/edeck/
TC_GEN_EDECK_INPUT_TEMPLATE = edeck*.dat

TC_GEN_SHAPE_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/TCGen_fcstGFS_
↪obsBDECK_2021season/shape
TC_GEN_SHAPE_INPUT_TEMPLATE = */{init?fmt=%Y}*/gtwo*.shp

TC_GEN_OUTPUT_DIR = {OUTPUT_BASE}/model_application/tc_and_extra_tc/TCGen
TC_GEN_OUTPUT_TEMPLATE = tc_gen


###
# TCGen Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcgen
###

TC_GEN_INIT_FREQ = 6

TC_GEN_VALID_FREQ = 6
```

(continues on next page)

(continued from previous page)

```
TC_GEN_FCST_HR_WINDOW_BEGIN = 6

TC_GEN_FCST_HR_WINDOW_END = 120

TC_GEN_MIN_DURATION = 12

TC_GEN_FCST_GENESIS_VMAX_THRESH = NA
TC_GEN_FCST_GENESIS_MSLP_THRESH = NA

TC_GEN_BEST_GENESIS_TECHNIQUE = BEST
TC_GEN_BEST_GENESIS_CATEGORY = TD, TS
TC_GEN_BEST_GENESIS_VMAX_THRESH = NA
TC_GEN_BEST_GENESIS_MSLP_THRESH = NA

TC_GEN_OPER_TECHNIQUE = CARQ

TC_GEN_DESC = ALL

MODEL = GFS

TC_GEN_DLAND_THRESH = NA

TC_GEN_GENESIS_MATCH_RADIUS = 500

TC_GEN_GENESIS_MATCH_POINT_TO_TRACK = True

TC_GEN_GENESIS_MATCH_WINDOW_BEG = -6
TC_GEN_GENESIS_MATCH_WINDOW_END = 6

TC_GEN_OPS_HIT_WINDOW_BEG = 0
TC_GEN_OPS_HIT_WINDOW_END = 48

TC_GEN_DEV_HIT_RADIUS = 500

TC_GEN_DEV_HIT_WINDOW_BEGIN = -24
TC_GEN_DEV_HIT_WINDOW_END = 24

TC_GEN_DISCARD_INIT_POST_GENESIS_FLAG = True

TC_GEN_DEV_METHOD_FLAG = True

TC_GEN_OPS_METHOD_FLAG = True

TC_GEN_CI_ALPHA = 0.05
```

(continues on next page)

(continued from previous page)

```
TC_GEN_OUTPUT_FLAG_FHO = NONE
TC_GEN_OUTPUT_FLAG_CTC = BOTH
TC_GEN_OUTPUT_FLAG_CTS = BOTH
TC_GEN_OUTPUT_FLAG_GENMPR = BOTH
TC_GEN_OUTPUT_FLAG_PCT = BOTH
TC_GEN_OUTPUT_FLAG_PSTD = BOTH
TC_GEN_OUTPUT_FLAG_PJC = BOTH
TC_GEN_OUTPUT_FLAG_PRC = BOTH

TC_GEN_NC_PAIRS_FLAG_LATLON = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_GENESIS = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_TRACKS  = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_FY_OY   = TRUE
TC_GEN_NC_PAIRS_FLAG_FCST_FY_ON   = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_GENESIS = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_TRACKS  = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_FY_OY   = TRUE
TC_GEN_NC_PAIRS_FLAG_BEST_FN_OY   = TRUE

TC_GEN_VALID_MINUS_GENESIS_DIFF_THRESH = NA

TC_GEN_BEST_UNIQUE_FLAG = TRUE

TC_GEN_DLAND_FILE = MET_BASE/tc_data/dland_global_tenth_degree.nc

TC_GEN_BASIN_FILE = MET_BASE/tc_data/basin_global_tenth_degree.nc

TC_GEN_NC_PAIRS_GRID = G003
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TC-Gen MET Configuration* (page 266) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// TC-Gen configuration file.
//
// For additional information, see the MET_BASE/config/README_TC file.
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

////////////////////////////////////////////////////////////////////////////////
//
// Genesis event definition criteria.
//
////////////////////////////////////////////////////////////////////////////////

//
// Model initialization frequency in hours, starting at 0.
//
// init_freq =
${METPLUS_INIT_FREQ}

//
// Valid hour frequency to be analyzed in hours, starting at 0
//
// valid_freq =
${METPLUS_VALID_FREQ}

//
// Forecast hours to be searched for genesis events
//
// fcst_hr_window =
${METPLUS_FCST_HR_WINDOW_DICT}

//
// Minimum track duration for genesis event in hours.
//
// min_duration =
${METPLUS_MIN_DURATION}

//
// Forecast genesis event criteria.  Defined as tracks reaching the specified
// intensity category, maximum wind speed threshold, and minimum sea-level
```

(continues on next page)

(continued from previous page)

```
// pressure threshold.  The forecast genesis time is the valid time of the first
// track point where all of these criteria are met.
//
// fcst_genesis =
${METPLUS_FCST_GENESIS_DICT}

//
// BEST track genesis event criteria.  Defined as tracks reaching the specified
// intensity category, maximum wind speed threshold, and minimum sea-level
// pressure threshold.  The BEST track genesis time is the valid time of the
// first track point where all of these criteria are met.
//
// best_genesis =
${METPLUS_BEST_GENESIS_DICT}

//
// Operational track technique name
//
// oper_technique =
${METPLUS_OPER_TECHNIQUE}

////////////////////////////////////////////////////////////////////////////////
//
// Track filtering options
// May be specified separately in each filter array entry.
//
////////////////////////////////////////////////////////////////////////////////

//
// Array of dictionaries containing the track filtering options
// If empty, a single filter is defined using the top-level settings.
//
// filter =
${METPLUS_FILTER}

//
// Description written to output DESC column
//
// desc =
${METPLUS_DESC}

//
// Forecast ATCF ID's
// If empty, all ATCF ID's found will be processed.
// Statistics will be generated separately for each ATCF ID.
```

(continues on next page)

(continued from previous page)

```
//
// model =
${METPLUS_MODEL}

//
// BEST and operational track storm identifiers
//
// storm_id =
${METPLUS_STORM_ID}

//
// BEST and operational track storm names
//
// storm_name =
${METPLUS_STORM_NAME}

//
// Forecast and operational initialization times to include or exclude
//
// init_beg =
${METPLUS_INIT_BEG}

// init_end =
${METPLUS_INIT_END}

// init_inc =
${METPLUS_INIT_INC}

// init_exc =
${METPLUS_INIT_EXC}

//
// Forecast, BEST, and operational valid time window
//
// valid_beg =
${METPLUS_VALID_BEG}

// valid_end =
${METPLUS_VALID_END}

//
// Forecast and operational initialization hours
//
// init_hour =
${METPLUS_INIT_HOUR}
```

(continues on next page)

(continued from previous page)

```
//
// Forecast and operational lead times in hours
//
// lead =
${METPLUS_LEAD}

//
// Spatial masking region (path to gridded data file or polyline file)
//
// vx_mask =
${METPLUS_VX_MASK}

//
// Spatial masking of hurricane basin names from the basin_file
//
// basin_mask =
${METPLUS_BASIN_MASK}

//
// Distance to land threshold
//
//dland_thresh =
${METPLUS_DLAND_THRESH}

////////////////////////////////////////////////////////////////////////////////
//
// Matching and scoring options
// May be specified separately in each filter array entry.
//
////////////////////////////////////////////////////////////////////////////////

//
// Genesis matching logic. Compare the forecast genesis point to all points in
// the Best track (TRUE) or the single Best track genesis point (FALSE).
//
//genesis_match_point_to_track =
${METPLUS_GENESIS_MATCH_POINT_TO_TRACK}

//
// Radius in km to search for a matching genesis event
//
// genesis_match_radius =
${METPLUS_GENESIS_MATCH_RADIUS}
```

(continues on next page)

(continued from previous page)

```
//
// Time window in hours, relative to the model genesis time, to search for a
// matching Best track point
//
//genesis_match_window = {
${METPLUS_GENESIS_MATCH_WINDOW_DICT}

//
// Radius in km for a development scoring method hit
//
// dev_hit_radius =
${METPLUS_DEV_HIT_RADIUS}

//
// Time window in hours for a development scoring method hit
//
// dev_hit_window =
${METPLUS_DEV_HIT_WINDOW_DICT}

// Time window in hours for the Best track genesis minus model initialization
// time difference for an operational scoring method hit
//
//ops_hit_window = {
${METPLUS_OPS_HIT_WINDOW_DICT}

//
// Discard genesis forecasts for initializations at or after the matching
// BEST track genesis time
//
// discard_init_post_genesis_flag =
${METPLUS_DISCARD_INIT_POST_GENESIS_FLAG}

//
// Scoring methods to be applied
//
//dev_method_flag =
${METPLUS_DEV_METHOD_FLAG}

// ops_method_flag =
${METPLUS_OPS_METHOD_FLAG}

////////////////////////////////////////////////////////////////////////////////
//
// Output options
// May be specified separately in each filter array entry.
```

(continues on next page)

(continued from previous page)

```
//
////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval alpha value
//
// ci_alpha =
${METPLUS_CI_ALPHA}

//
// Statistical output types
//
// output_flag =
${METPLUS_OUTPUT_FLAG_DICT}

//
// NetCDF genesis pair counts
//
// nc_pairs_flag =
${METPLUS_NC_PAIRS_FLAG_DICT}

//
// Specify which track points should be counted by thresholding the track point
// valid time minus genesis time difference.
//
// valid_minus_genesis_diff_thresh =
${METPLUS_VALID_MINUS_GENESIS_DIFF_THRESH}

//
// Count unique BEST track genesis event locations (TRUE) versus counting the
// location for all pairs (FALSE).
//
// best_unique_flag =
${METPLUS_BEST_UNIQUE_FLAG}

////////////////////////////////////////////////////////////////////////////////
//
// Global settings
// May only be specified once.
//
////////////////////////////////////////////////////////////////////////////////

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
```

(continues on next page)

(continued from previous page)

```
//
// dland_file =
${METPLUS_DLAND_FILE}

//
// Specify the NetCDF file containing a gridded representation of the
// global basins.
//
// basin_file =
${METPLUS_BASIN_FILE}

//
// NetCDF genesis pairs grid
//
// nc_pairs_grid =
${METPLUS_NC_PAIRS_GRID}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V10.0.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in the use case configuration file then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/model_applications/tc_and_extra_tc/TCGen_
↪fcstGFS_obsBDECK_2021season.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in use case configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/tc_and_extra_tc/
↪TCGen_fcstGFS_obsBDECK_2021season.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in

parm/use_cases

- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in model_applications/tc_and_extra_tc/TCGen (relative to **OUTPUT_BASE**) and will contain the following files:

- tc_gen.stat
- tc_gen_pstd.txt
- tc_gen_prc.txt
- tc_gen_pjc.txt
- tc_gen_pct.txt
- tc_gen_cts.txt
- tc_gen_ctc.txt
- tc_gen_genmpr.txt
- tc_gen_pairs.nc

Keywords

Note:

- TCGenToolUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/tc_and_extra_tc-TCGen_fcstGFS_obsBDECK_2021season.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.15.7 CycloneVerification: TC Verification Compare ADECK vs BDECK

model_applications/tc_and_extra_tc/TCPairs_TCStat_fcstADECK_obsBDECK_ATCF_BasicExample.conf

Scientific Objective

This use case run TC-Pairs to produce produce matched pairs of forecast model output and an observation dataset. TC-Pairs produces matched pairs for position errors, as well as wind, sea level pressure, and distance to land values for each input dataset. Then TC-stat will filter TC-pairs output based on user criteria.

Datasets

Forecast: Adeck

/path/to/TCPairs_TCStat_fcstADECK_obsBDECK_ATCF_BasicExample/a{basin}{cyclone}{init?fmt=%Y}.dat

Observation: Bdeck

/path/to/{TCPairs_TCStat_fcstADECK_obsBDECK_ATCF_BasicExample/b{basin}{cyclone}{init?fmt=%Y}.dat

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1973) section for more information.

Data Source: NHC ftp.noaa.gov/atcf

METplus Workflow

The following tools are used for each run time:

TCPairs TCStat

To generate TCPairs output, this example loops by initialization time for every 6 hour period that is available in the data set between 2021082500 and 2021083000. Then TCStat filters the TCPairs output based on user criteria (e.g. storm characteristics in this use case).

METplus Components

This use case first runs TC-Pairs to produce matched pairs of Adeck and Bdeck files. The TC-Pairs output (tcst files) is then read by the TC-Stat tool to further filter the tcst files as well as summarize the statistical information.

METplus Workflow

TCPairs is the first tool called in this example. It processes the following run times for each storm file (e.g. aal092021.dat, aal102021.dat) against the corresponding Bdeck files (e.g. bal092021.dat, bal102021.dat):

Init/Valid: 2021082500
End/Valid: 2021083000

TC-Stat is the second (and final) tool called in this example. It processes the output from TCPairs. In this example the TC-Stat filters the TC-Pairs output based on the characteristics of the storm (HU, SD, SS, TS, TD). The output from the TC-Stat can be used to aggregate verification statistics (e.g. Track, Intensity, MSLP, wind radii errors etc.).

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c /path/to/TCPairs_TCStat_fcstADECK_obsBDECK_ATCF_BasicExample.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/tc_and_extra_tc/
↪TCPairs_TCStat_fcstADECK_obsBDECK_ATCF_BasicExample.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide
```

(continues on next page)

(continued from previous page)

```
###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCPairs, TCStat


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2021082500
INIT_END = 2021083000
INIT_INCREMENT = 21600

###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

# TCPairs

TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/TCPairs_TCStat_
↪fcstADECK_obsBDECK_ATCF_BasicExample
TC_PAIRS_BDECK_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/TCPairs_TCStat_
↪fcstADECK_obsBDECK_ATCF_BasicExample

TC_PAIRS_ADECK_TEMPLATE = a{basin}{cyclone}{init?fmt=%Y}.dat
TC_PAIRS_BDECK_TEMPLATE = b{basin}{cyclone}{init?fmt=%Y}.dat

TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs
TC_PAIRS_OUTPUT_TEMPLATE = tc_pairs.{basin}{cyclone}{init?fmt=%Y}
```

(continues on next page)

(continued from previous page)

```
# TCStat

TC_STAT_LOOKIN_DIR = {TC_PAIRS_OUTPUT_DIR}
TC_STAT_OUTPUT_DIR = {OUTPUT_BASE}/tc_stat


###
# TCPairs Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcpairs
###

MODEL = OFCL, HWRF

TC_PAIRS_BASIN = AL
TC_PAIRS_CYCLONE = 09, 10

TC_PAIRS_DLAND_FILE = MET_BASE/tc_data/dland_global_tenth_degree.nc

TC_PAIRS_MATCH_POINTS = TRUE


###
# TCStat Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcstat
###

TC_STAT_LINE_TYPE = TCMPR

TC_STAT_COLUMN_STRING_NAME = LEVEL
TC_STAT_COLUMN_STRING_VAL = HU,SD,SS,TS,TD

TC_STAT_WATER_ONLY = FALSE

TC_STAT_JOB_ARGS = -job filter -dump_row {TC_STAT_OUTPUT_DIR}/tc_stat_summary.tcst
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TCPairs MET Configuration* (page 282) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

//
// Models
//
${METPLUS_MODEL}

//
// Description
//
${METPLUS_DESC}

//
// Storm identifiers
//
${METPLUS_STORM_ID}

//
// Basins
//
${METPLUS_BASIN}

//
// Cyclone numbers
//
${METPLUS_CYCLONE}

//
// Storm names
//
${METPLUS_STORM_NAME}

//
```

(continues on next page)

(continued from previous page)

```
// Model initialization time windows to include or exclude
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
${METPLUS_VALID_INC}
// valid_exc =
${METPLUS_VALID_EXC}

// write_valid =
${METPLUS_WRITE_VALID}

//
// Valid model time window
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}

//
// Model initialization hours
//
init_hour = [];

//
// Required lead time in hours
//
lead_req = [];

//
// lat/lon polylines defining masking regions
//
init_mask  = "";
valid_mask = "";

//
// Specify if the code should check for duplicate ATCF lines
//
//check_dup =
${METPLUS_CHECK_DUP}
```

(continues on next page)

(continued from previous page)

```
//
// Specify special processing to be performed for interpolated models.
// Set to NONE, FILL, or REPLACE.
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
//
//consensus =
${METPLUS_CONSENSUS_LIST}


//
// Forecast lag times
//
lag_time = [];

//
// CLIPER/SHIFOR baseline forecasts to be derived from the BEST
// and operational (CARQ) tracks.
//
best_technique = [ "BEST" ];
best_baseline  = [];
oper_technique = [ "CARQ" ];
oper_baseline  = [];

//
// Specify the datasets to be searched for analysis tracks (NONE, ADECK, BDECK,
// or BOTH).
//
anly_track = BDECK;

//
// Specify if only those track points common to both the ADECK and BDECK
// tracks be written out.
//
//match_points =
${METPLUS_MATCH_POINTS}

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
```

(continues on next page)

(continued from previous page)

```
//
${METPLUS_DLAND_FILE}

//
// Specify watch/warning information:
//   - Input watch/warning filename
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}


//diag_info_map = {
${METPLUS_DIAG_INFO_MAP_LIST}

//diag_convert_map = {
${METPLUS_DIAG_CONVERT_MAP_LIST}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCStat configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// The parameters listed below are used to filter the TC-STAT data down to the
// desired subset of lines over which statistics are to be computed.  Only
// those lines which meet ALL of the criteria specified will be retained.
//
// The settings that are common to all jobs may be specified once at the top
// level.  If no selection is listed for a parameter, that parameter will not
// be used for filtering.  If multiple selections are listed for a parameter,
// the analyses will be performed on their union.
```

(continues on next page)

(continued from previous page)

```
//

//
// Stratify by the AMODEL or BMODEL columns.
//
${METPLUS_AMODEL}
${METPLUS_BMODEL}

//
// Stratify by the DESC column.
//
${METPLUS_DESC}

//
// Stratify by the STORM_ID column.
//
${METPLUS_STORM_ID}

//
// Stratify by the BASIN column.
// May add using the "-basin" job command option.
//
${METPLUS_BASIN}

//
// Stratify by the CYCLONE column.
// May add using the "-cyclone" job command option.
//
${METPLUS_CYCLONE}

//
// Stratify by the STORM_NAME column.
// May add using the "-storm_name" job command option.
//
${METPLUS_STORM_NAME}

//
// Stratify by the INIT times.
// Model initialization time windows to include or exclude
// May modify using the "-init_beg", "-init_end", "-init_inc",
// and "-init_exc" job command options.
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
${METPLUS_INIT_INC}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_INIT_EXC}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INC}
${METPLUS_VALID_EXC}

//
// Stratify by the initialization and valid hours and lead time.
//
${METPLUS_INIT_HOUR}
${METPLUS_VALID_HOUR}
${METPLUS_LEAD}

//
// Select tracks which contain all required lead times.
//
${METPLUS_LEAD_REQ}

//
// Stratify by the INIT_MASK and VALID_MASK columns.
//
${METPLUS_INIT_MASK}
${METPLUS_VALID_MASK}


//
// Stratify by the LINE_TYPE column.
//
//line_type =
${METPLUS_LINE_TYPE}


//
// Stratify by checking the watch/warning status for each track point
// common to both the ADECK and BDECK tracks.  If the watch/warning status
// of any of the track points appears in the list, retain the entire track.
//
${METPLUS_TRACK_WATCH_WARN}

//
// Stratify by applying thresholds to numeric data columns.
```

(continues on next page)

(continued from previous page)

```
//
${METPLUS_COLUMN_THRESH_NAME}
${METPLUS_COLUMN_THRESH_VAL}

//
// Stratify by performing string matching on non-numeric data columns.
//
${METPLUS_COLUMN_STR_NAME}
${METPLUS_COLUMN_STR_VAL}

//
// Stratify by excluding strings in non-numeric data columns.
//
//column_str_exc_name =
${METPLUS_COLUMN_STR_EXC_NAME}

//column_str_exc_val =
${METPLUS_COLUMN_STR_EXC_VAL}

//
// Similar to the column_thresh options above
//
${METPLUS_INIT_THRESH_NAME}
${METPLUS_INIT_THRESH_VAL}

//
// Similar to the column_str options above
//
${METPLUS_INIT_STR_NAME}
${METPLUS_INIT_STR_VAL}

//
// Similar to the column_str_exc options above
//
//init_str_exc_name =
${METPLUS_INIT_STR_EXC_NAME}

//init_str_exc_val =
${METPLUS_INIT_STR_EXC_VAL}

//diag_thresh_name =
${METPLUS_DIAG_THRESH_NAME}

//diag_thresh_val =
${METPLUS_DIAG_THRESH_VAL}
```

(continues on next page)

(continued from previous page)

```
//init_diag_thresh_name =
${METPLUS_INIT_DIAG_THRESH_NAME}

//init_diag_thresh_val =
${METPLUS_INIT_DIAG_THRESH_VAL}


//
// Stratify by the ADECK and BDECK distances to land.
//
${METPLUS_WATER_ONLY}

//
// Specify whether only those track points occurring near landfall should be
// retained, and define the landfall retention window in HH[MMSS] format
// around the landfall time.
//
${METPLUS_LANDFALL}
${METPLUS_LANDFALL_BEG}
${METPLUS_LANDFALL_END}

//
// Specify whether only those track points common to both the ADECK and BDECK
// tracks should be retained.  May modify using the "-match_points" job command
// option.
//
${METPLUS_MATCH_POINTS}

//event_equal =
${METPLUS_EVENT_EQUAL}

//event_equal_lead =
${METPLUS_EVENT_EQUAL_LEAD}

//out_init_mask =
${METPLUS_OUT_INIT_MASK}

//out_valid_mask =
${METPLUS_OUT_VALID_MASK}


//
// Array of TCStat analysis jobs to be performed on the filtered data
//
```

(continues on next page)

(continued from previous page)

```
${METPLUS_JOBS}

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

It is recommended to run this use case by:

Passing in TCPairs_TCStat_fcstADECK_obsBDECK_ATCF_BasicExample.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/TCPairs_TCStat_fcstADECK_obsBDECK_ATCF_BasicExample.conf -c /path/
↪to/user_system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where Adeck and Bdeck ATCF format files are read (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in tc_pairs/ tc_stat/ (relative to **OUTPUT_BASE**) and will contain the following files:

- tc_pairs/tc_pairs.al092021.tcst
- tc_pairs/tc_pairs.al102021.tcst
- tc_stat/tc_stat_summary.tcst

Keywords

Note:

- TCPairsToolUseCase
- TCStatToolUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.15.8 Cyclone Plotter: From TC-Pairs Output

model_applications/tc_and_extra_tc/Plotter_fcstGFS _obsGFS_ExtraTC.conf

Scientific Objective

Provide visualization of storm tracks using output from the MET TC-Pairs tool. The date and hour associated with each storm track indicates the first time the storm was tracked in the model.

Datasets

- Forecast dataset: ADeck modified-ATCF tropical cyclone data
- Observation dataset: BDeck modified-ATCF "best-track" tropical cyclone data

METplus Components

This use case first runs TCPairs and then generates the storm track plot for all storm tracks found in the .tcst output file created by the MET TC-Pairs tool.

METplus Workflow

The following tools are used for each run time:

TCPairs

To generate TCPairs output, this example loops by initialization time for every 6 hour period that is available in the data set for 20150301. The output is then used to generate the plot of all cyclone tracks.

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/tc_and_extra_tc/Plotter_fcstGFS_obsGFS_ExtraTC.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/tc_and_extra_tc/
↪Plotter_fcstGFS_obsGFS_ExtraTC.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = TCPairs, CyclonePlotter


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = init
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20150301
INIT_END = 20150330
INIT_INCREMENT = 21600

TC_PAIRS_RUN_ONCE = True


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
```

(continues on next page)

(continued from previous page)

```
↪and-filename-template-info
###

# TCPairs

TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/track_data
TC_PAIRS_ADECK_TEMPLATE = {date?fmt=%Y%m}/a{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}

TC_PAIRS_BDECK_INPUT_DIR = {TC_PAIRS_ADECK_INPUT_DIR}
TC_PAIRS_BDECK_TEMPLATE = {date?fmt=%Y%m}/b{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}

TC_PAIRS_REFORMAT_DIR = {OUTPUT_BASE}/track_data_atcf
TC_PAIRS_SKIP_IF_REFORMAT_EXISTS = yes

TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs
TC_PAIRS_OUTPUT_TEMPLATE = {date?fmt=%Y%m}/{basin?fmt=%s}q{date?fmt=%Y%m%d%H}.gfso.{cyclone?
↪fmt=%s}
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = yes


# CyclonePlotter

CYCLONE_PLOTTER_INPUT_DIR = {OUTPUT_BASE}/tc_pairs
CYCLONE_PLOTTER_OUTPUT_DIR = {OUTPUT_BASE}/cyclone


###
# TCPairs Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#tcpairs
###

TC_PAIRS_DLAND_FILE = {MET_INSTALL_DIR}/share/met/tc_data/dland_global_tenth_degree.nc

TC_PAIRS_REFORMAT_DECK = yes
TC_PAIRS_REFORMAT_TYPE = SBU

TC_PAIRS_MISSING_VAL_TO_REPLACE = -99
TC_PAIRS_MISSING_VAL = -9999


###
# CyclonePlotter Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#cycloneplotter
```

(continues on next page)

(continued from previous page)

```
###

CYCLONE_PLOTTER_INIT_DATE = 20150301

CYCLONE_PLOTTER_INIT_HR = 12 ;; hh format
CYCLONE_PLOTTER_MODEL = GFSO
CYCLONE_PLOTTER_PLOT_TITLE = Model Forecast Storm Tracks

CYCLONE_PLOTTER_GLOBAL_PLOT = no

# ***IMPORTANT*** If CYCLONE_PLOTTER_GLOBAL_PLOT
# is set to False or N[n]o, then define the region of the world to plot.
# Longitudes can range from -180 to 180 degrees and latitudes from -90 to 90 degrees

# --------------------------------
# EXAMPLE OF BOUNDING BOX SETTINGS
# --------------------------------
# NORTHERN HEMISPHERE
CYCLONE_PLOTTER_WEST_LON = -180
CYCLONE_PLOTTER_EAST_LON = 179
CYCLONE_PLOTTER_SOUTH_LAT = 0
CYCLONE_PLOTTER_NORTH_LAT = 90


CYCLONE_PLOTTER_CIRCLE_MARKER_SIZE = 2
CYCLONE_PLOTTER_CROSS_MARKER_SIZE = 3

CYCLONE_PLOTTER_ANNOTATION_FONT_SIZE = 3
CYCLONE_PLOTTER_LEGEND_FONT_SIZE = 3

CYCLONE_PLOTTER_GENERATE_TRACK_ASCII = yes

CYCLONE_PLOTTER_ADD_WATERMARK = False

CYCLONE_PLOTTER_RESOLUTION_DPI = 400
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *TCPairs MET Configuration* (page 282) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

//
// Models
//
${METPLUS_MODEL}

//
// Description
//
${METPLUS_DESC}

//
// Storm identifiers
//
${METPLUS_STORM_ID}

//
// Basins
//
${METPLUS_BASIN}

//
// Cyclone numbers
//
${METPLUS_CYCLONE}

//
// Storm names
//
${METPLUS_STORM_NAME}

//
```

(continues on next page)

(continued from previous page)

```
// Model initialization time windows to include or exclude
//
${METPLUS_INIT_BEG}
${METPLUS_INIT_END}
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
${METPLUS_VALID_INC}
// valid_exc =
${METPLUS_VALID_EXC}

// write_valid =
${METPLUS_WRITE_VALID}

//
// Valid model time window
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}

//
// Model initialization hours
//
init_hour = [];

//
// Required lead time in hours
//
lead_req = [];

//
// lat/lon polylines defining masking regions
//
init_mask  = "";
valid_mask = "";

//
// Specify if the code should check for duplicate ATCF lines
//
//check_dup =
${METPLUS_CHECK_DUP}
```

(continues on next page)

(continued from previous page)

```
//
// Specify special processing to be performed for interpolated models.
// Set to NONE, FILL, or REPLACE.
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
//
//consensus =
${METPLUS_CONSENSUS_LIST}


//
// Forecast lag times
//
lag_time = [];

//
// CLIPER/SHIFOR baseline forecasts to be derived from the BEST
// and operational (CARQ) tracks.
//
best_technique = [ "BEST" ];
best_baseline  = [];
oper_technique = [ "CARQ" ];
oper_baseline  = [];

//
// Specify the datasets to be searched for analysis tracks (NONE, ADECK, BDECK,
// or BOTH).
//
anly_track = BDECK;

//
// Specify if only those track points common to both the ADECK and BDECK
// tracks be written out.
//
//match_points =
${METPLUS_MATCH_POINTS}

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
```

(continues on next page)

(continued from previous page)

```
//
${METPLUS_DLAND_FILE}

//
// Specify watch/warning information:
//   - Input watch/warning filename
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}


//diag_info_map = {
${METPLUS_DIAG_INFO_MAP_LIST}

//diag_convert_map = {
${METPLUS_DIAG_CONVERT_MAP_LIST}

//
// Indicate a version number for the contents of this configuration file.
// The value should generally not be modified.
//
//version = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Running METplus

This use case can be run two ways:

1) Passing in Plotter_fcstGFS_obsGFS_ExtraTC.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/tc_and_extra_tc/
↪Plotter_fcstGFS_obsGFS_ExtraTC.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in Plotter_fcstGFS_obsGFS_ExtraTC.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/tc_and_extra_tc/
↪Plotter_fcstGFS_obsGFS_ExtraTC.conf
```

The former method is recommended. Whether you add them to a user-specific configuration file or modify

the metplus_config files, the following variables must be set correctly:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs). This is not required to run METplus, but it is required to run the examples in parm/use_cases
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions
- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will generate the following output to both the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Additionally, two output files are created. Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. TCPairs output for this use case will be found in tc_pairs/201503 (relative to **OUTPUT_BASE**) and will contain files with the following format:

- mlq2015030100.gfso.<*nnnn*>.tcst

where *nnnn* is a zero-padded 4-digit number

A plot (in .png format) will be found in the cyclone directory (relative to **OUTPUT_BASE**) along with a text file containing data corresponding to the plotted storm tracks:

- 20150301.png
- 20150301.txt

Keywords

Note:

- TCPairsToolUseCase
- CyclonePlotterUseCase
- FeatureRelativeUseCase
- MediumRangeAppUseCase

- NOAAEMCOrgUseCase
- SBUOrgUseCase
- DTCOrgUseCase
- TropicalCycloneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/tc_and_extra_tc-Plotter_fcstGFS_obsGFS_ExtraTC.png'

Total running time of the script: (0 minutes 0.000 seconds)

7.2.16.16 Unstructured Grids

Unstructured grids used by models for numerical weather prediction.

7.2.16.16.1 StatAnalysis: Met Office LFRic UGRID

model_applications/unstructured_grids/StatAnalysis_fcstLFRIC_UGRID_obsASCII_PyEmbed.conf

Scientific Objective

This use case demonstrates the use of python embedding to ingest and perform verification on an unstructured grid. This foregoes the need to interpolate to a regular grid as a step in the verification process, thereby avoiding any incurred interpolation error in the process.

In particular, this use case ingests a UK MET Office LFRic forecast file in NetCDF format, which resides in the UGRID format of the cubed-sphere. The python library Iris was developed to perform analysis on various UGRID formats, and is employed here to ingest the file as well as perform direct interpolation from the native forecast grid to observation locations, thereby forming matched pairs to pass to stat_analysis. In order to perform the interpolation using a nearest-neighbors approach, the geovista python package is also used to form a KD tree to be used in identifying the interpolation points to be used. This package is located at https://github.com/bjlittle/geovista/ and can be installed from a development version. It is also required to install the pyvista python package. ASCII files containing observations are also ingested.

The python embedding script itself performs the interpolation in time, and for this use case thins the observation data in order to reduce the run time. It is also noted that the observations for this use case were fabricated and correlated observation-forecast pairs are not expected.

Datasets

Data source: UK MET Office LFRic forecast files in UGRID NetCDF format and observations in ASCII format

Location: All of the input data required for this use case can be found in the met_test sample data tarball. Click here to the METplus releases page and download sample data for the appropriate release: https://github.com/dtcenter/METplus/releases
The tarball should be unpacked into the directory that you will set the value of INPUT_BASE. See *Running METplus* (page 1994) section for more information.

METplus Components

This use case utilizes the METplus StatAnalysis wrapper to search for files that are valid for the given case and generate a command to run the MET tool stat_analysis.

METplus Workflow

StatAnalysis is the only tool called in this example. It processes the following run times:

Valid: 2021-05-05_00Z
Forecast lead: 12 hour

METplus Configuration

METplus first loads all of the configuration files found in parm/metplus_config, then it loads any configuration files passed to METplus via the command line with the -c option, i.e. -c parm/use_cases/model_applications/unstructured_grids/StatAnalysis_fcstLFRIC_UGRID_obsASCII_PyEmbed.conf

```
[config]

# Documentation for this use case can be found at
# https://metplus.readthedocs.io/en/latest/generated/model_applications/unstructured_grids/
↪StatAnalysis_fcstLFRIC_UGRID_obsASCII_PyEmbed.html

# For additional information, please see the METplus Users Guide.
# https://metplus.readthedocs.io/en/latest/Users_Guide

###
# Processes to run
```

(continues on next page)

(continued from previous page)

```
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#process-list
###

PROCESS_LIST = StatAnalysis


###
# Time Info
# LOOP_BY options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
#   INIT_TIME_FMT, INIT_BEG, INIT_END, and INIT_INCREMENT must also be set
# If set to VALID or REALTIME:
#   VALID_TIME_FMT, VALID_BEG, VALID_END, and VALID_INCREMENT must also be set
# LEAD_SEQ is the list of forecast leads to process
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#timing-
↪control
###

LOOP_BY = VALID

VALID_TIME_FMT = %Y%m%d%H
VALID_BEG=2021050500
VALID_END=2021050500
VALID_INCREMENT = 6H

LEAD_SEQ = 0


###
# File I/O
# https://metplus.readthedocs.io/en/latest/Users_Guide/systemconfiguration.html#directory-
↪and-filename-template-info
###

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = python {PARM_BASE}/use_cases/model_applications/
↪unstructured_grids/StatAnalysis_fcstLFRIC_UGRID_obsASCII_PyEmbed/ugrid_lfric_mpr.py {INPUT_
↪BASE}/model_applications/unstructured_grids/StatAnalysis_fcstLFRIC_UGRID_obsASCII_PyEmbed/
↪fcst_data/lfric_ver_20210505_0000.nc {INPUT_BASE}/model_applications/unstructured_grids/
↪StatAnalysis_fcstLFRIC_UGRID_obsASCII_PyEmbed/obs_data

STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/StatAnalysis_UGRID
STAT_ANALYSIS_OUTPUT_TEMPLATE = job.out
MODEL1_STAT_ANALYSIS_DUMP_ROW_TEMPLATE = dump.out
```

(continues on next page)

(continued from previous page)

```
###
# StatAnalysis Settings
# https://metplus.readthedocs.io/en/latest/Users_Guide/wrappers.html#statanalysis
###

MODEL1 = NA
MODEL1_OBTYPE = NA

STAT_ANALYSIS_JOB_NAME = aggregate_stat
STAT_ANALYSIS_JOB_ARGS = -out_line_type CNT -dump_row [dump_row_file] -line_type MPR -by␣
↪FCST_VAR

MODEL_LIST =
DESC_LIST =
FCST_LEAD_LIST =
OBS_LEAD_LIST =
FCST_VALID_HOUR_LIST =
FCST_INIT_HOUR_LIST =
OBS_VALID_HOUR_LIST =
OBS_INIT_HOUR_LIST =
FCST_VAR_LIST =
OBS_VAR_LIST =
FCST_UNITS_LIST =
OBS_UNITS_LIST =
FCST_LEVEL_LIST =
OBS_LEVEL_LIST =
VX_MASK_LIST =
INTERP_MTHD_LIST =
INTERP_PNTS_LIST =
FCST_THRESH_LIST =
OBS_THRESH_LIST =
COV_THRESH_LIST =
ALPHA_LIST =
LINE_TYPE_LIST =

GROUP_LIST_ITEMS =
LOOP_LIST_ITEMS = MODEL_LIST
```

MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 74) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVERWRITTEN BY METPLUS WHEN IT CALLS THE MET TOOLS!

If there is a setting in the MET configuration file that is currently not supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 87)

Note: See the *StatAnalysis MET Configuration* (page 246) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// STAT-Analysis configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

//
// Filtering input STAT lines by the contents of each column
//
${METPLUS_MODEL}
${METPLUS_DESC}

${METPLUS_FCST_LEAD}
${METPLUS_OBS_LEAD}

${METPLUS_FCST_VALID_BEG}
${METPLUS_FCST_VALID_END}
${METPLUS_FCST_VALID_HOUR}

${METPLUS_OBS_VALID_BEG}
${METPLUS_OBS_VALID_END}
${METPLUS_OBS_VALID_HOUR}

${METPLUS_FCST_INIT_BEG}
${METPLUS_FCST_INIT_END}
${METPLUS_FCST_INIT_HOUR}

${METPLUS_OBS_INIT_BEG}
${METPLUS_OBS_INIT_END}
${METPLUS_OBS_INIT_HOUR}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_FCST_VAR}
${METPLUS_OBS_VAR}

${METPLUS_FCST_UNITS}
${METPLUS_OBS_UNITS}

${METPLUS_FCST_LEVEL}
${METPLUS_OBS_LEVEL}

${METPLUS_OBTYPE}

${METPLUS_VX_MASK}

${METPLUS_INTERP_MTHD}

${METPLUS_INTERP_PNTS}

${METPLUS_FCST_THRESH}
${METPLUS_OBS_THRESH}
${METPLUS_COV_THRESH}

${METPLUS_ALPHA}

${METPLUS_LINE_TYPE}

column = [];

weight = [];

////////////////////////////////////////////////////////////////////////////////

//
// Array of STAT-Analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval settings
//
out_alpha = 0.05;

boot = {
```

(continues on next page)

(continued from previous page)

```
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

////////////////////////////////////////////////////////////////////////////////

//
// WMO mean computation logic
//
wmo_sqrt_stats   = [ "CNT:FSTDEV",  "CNT:OSTDEV",  "CNT:ESTDEV",
                     "CNT:RMSE",    "CNT:RMSFA",   "CNT:RMSOA",
                     "VCNT:FS_RMS", "VCNT:OS_RMS", "VCNT:RMSVE",
                     "VCNT:FSTDEV", "VCNT:OSTDEV" ];

wmo_fisher_stats = [ "CNT:PR_CORR", "CNT:SP_CORR",
                     "CNT:KT_CORR", "CNT:ANOM_CORR" ];

////////////////////////////////////////////////////////////////////////////////

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;
vif_flag       = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

${METPLUS_MET_CONFIG_OVERRIDES}
```

Python Embedding

This use case uses a Python embedding script to read input data

parm/use_cases/model_applications/unstructured_grids/StatAnalysis_fcstLFRIC_UGRID_obsASCII_PyEmbed/ugrid_lf

```
from __future__ import print_function

import math
import pandas as pd
import numpy as np
import os
```

(continues on next page)

(continued from previous page)

```
from glob import glob
import sys
import xarray as xr
import datetime as dt
import iris
from iris.experimental.ugrid import PARSE_UGRID_ON_LOAD
#geovista from https://github.com/bjlittle/geovista/
import geovista as gv
import geovista.theme
from geovista.common import to_xyz
import netCDF4
import pyvista as pv
from pykdtree.kdtree import KDTree

from pathlib import Path
from typing import Optional

import matplotlib.pyplot as plt

print(f"{iris.__version__=}")
print(f"{gv.__version__=}")

################################################################################

def read_ascii_obs(files):
    paths = sorted(glob(files))
    datasets = [pd.read_table(p, header=None, delim_whitespace=True) for p in paths]
    combined = pd.concat(datasets)
    return combined

def load_ugrid(
    fname: str,
    data: Optional[bool] = False,
    constraint: Optional[str] = None,
    verbose: Optional[bool] = False
) -> pv.PolyData:
#    fname = BASE_DIR / fname
    with PARSE_UGRID_ON_LOAD.context():
        cube = iris.load_cube(fname, constraint=constraint)

    if cube.ndim > 1:
        cube = cube[(0,) * (cube.ndim - 1)]

    if verbose:
        print(cube)
```

(continues on next page)

(continued from previous page)

```
    data = cube.data if data else None

    face_node = cube.mesh.face_node_connectivity
    indices = face_node.indices_by_location()
    lons, lats = cube.mesh.node_coords

    mesh = gv.Transform.from_unstructured(
        lons.points,
        lats.points,
        indices,
        data=data,
        start_index=face_node.start_index,
        name=cube.name(),
    )

    if data is None:
        mesh.active_scalars_name = None

    return mesh

def info(mesh: pv.PolyData) -> None:
    print(f"The mesh is a C{int(math.sqrt(mesh.n_cells / 6))}, with 6 panels, {int(mesh.n_
↪cells / 6):,d} cells per panel, and {mesh.n_cells:,d} cells.")

def find_nearest(tree, points, poi, k):
    # lat/lon to xyz
    xyz = to_xyz(*poi)

    # find the k nearest euclidean neighbours
    dist, idxs = tree.query(xyz, k=k)

    if idxs.ndim > 1:
        idxs = idxs[0]

    # retieve the associated xyz points of the k nearest neighbours
    nearest = points[idxs]

    return xyz, nearest, idxs

def to_centers(mesh: pv.PolyData) -> pv.PolyData:
    tmp = mesh.copy()
    tmp.clear_cell_data()
    tmp.clear_point_data()
    tmp.clear_field_data()
```

(continues on next page)

(continued from previous page)

```
    return tmp.cell_centers()

#########################################################################
print('Python Script:\t', sys.argv[0])

# Input is directory of .nc lfric files and a directory of ascii obs filess

if len(sys.argv) == 3:
    # Read the input file as the first argument
    input_fcst_dir = os.path.expandvars(sys.argv[1])
    input_obs_dir = os.path.expandvars(sys.argv[2])
    try:
        print("Input Forecast Dir:\t" + repr(input_fcst_dir))
        print("Input Observations Dir:\t" + repr(input_obs_dir))

        #Read all obs from directory
        obs_data = read_ascii_obs(input_obs_dir+'/*.ascii')
        print(obs_data.shape)
        obs_data = obs_data.iloc[::1000, :]#thin for testing
        obs_data = obs_data.rename(columns={0:'message_type', 1:'station_id', 2:'obs_valid_
↪time', 3:'obs_lat', 4:'obs_lon', \
                                5:'elevation', 6:'var_name', 7:'level', 8:'height', 9:'qc_string
↪', 10:'obs_value'})

        obs_vars = ['UGRD', 'VGRD', 'TMP', 'RH']
        fcst_vars = ['u10m', 'v10m', 't1p5m', 'rh1p5m']

        #open the netcdf forecast to access data values and list of times
        fcst_data = xr.open_dataset(input_fcst_dir)
        fcst_times = pd.to_datetime(fcst_data.coords['time_centered'])

        match_df = pd.DataFrame(columns=['message_type', 'station_id', 'obs_valid_time',
↪'obs_lat', 'obs_lon', \
                'elevation', 'var_name', 'level', 'height', 'qc_string', 'obs_value', 'idx_
↪nearest, fcst_value'])

        for idx1, (obs_var, fcst_var) in enumerate(zip(obs_vars, fcst_vars)):

            #load forecast as an iris cube
            fcst_mesh = load_ugrid(input_fcst_dir, constraint=fcst_var)
            info(fcst_mesh)

            #get indices of nearest cell center
            fcst_centers = to_centers(fcst_mesh)
            points = fcst_centers.points
```

(continues on next page)

(continued from previous page)

```
            tree = KDTree(points)

            #get the forecast data values loaded
            fcst_df = fcst_data[fcst_var].to_dataframe()
            print(fcst_df)

            #get obs data for variable
            var_data = obs_data.loc[obs_data['var_name'] == obs_var].reset_index(drop=True)

            for idx2, row in var_data.iterrows():
                xyz, nearest, idx_nearest = find_nearest(tree, points, [row['obs_lat'], row[
↪'obs_lon']], k=1)
                var_data.at[idx2,'idx_nearest'] = int(idx_nearest)

                #get the obs time, search for closest in the forecast data
                time = dt.datetime.strptime(row['obs_valid_time'],'%Y%m%d_%H%M%S')
                match_time = min(fcst_times, key=lambda  d: abs(d - time))
                match_idx = np.argmin(np.abs(fcst_times - time))

                #add matched fcst value to data
                var_data.at[idx2, 'fcst_value'] = fcst_df.loc[(match_idx,int(idx_nearest)),␣
↪fcst_var]
                var_data.at[idx2, 'fcst_lat'] = fcst_df.loc[(match_idx,int(idx_nearest)),
↪'Mesh2d_face_x']
                var_data.at[idx2, 'fcst_lon'] = fcst_df.loc[(match_idx,int(idx_nearest)),
↪'Mesh2d_face_y']
                var_data.at[idx2, 'fcst_time'] = fcst_df.loc[(match_idx,int(idx_nearest)),
↪'time_centered']

            #check results
            #with pd.option_context('display.max_rows', None):
            #    print(var_data[['obs_lat','fcst_lat','obs_lon','fcst_lon','obs_value','fcst_
↪value','obs_valid_time','fcst_time']])
            with pd.option_context('display.max_columns', 500, 'display.max_rows', 100,
↪'display.width', 500):
                print(var_data)
            ob_vals = var_data['obs_value'].values
            f_vals = var_data['fcst_value'].values

            match_df = pd.concat([match_df, var_data], ignore_index=True)

        nlocs = len(match_df.index)
        print('Number of locations in matched set: ' + str(nlocs))

        # Add additional columns
```

(continues on next page)

(continued from previous page)

```
        match_df['lead'] = '000000'
        match_df['MPR'] = 'MPR'
        match_df['nobs'] = nlocs
        match_df['index'] = range(0,nlocs)
        match_df['na'] = 'NA'
        match_df['QC'] = '0'

        # Arrange columns in MPR format
        cols = ['na','na','lead','obs_valid_time','obs_valid_time','lead','obs_valid_time',
                'obs_valid_time','var_name','na','lead','var_name','na','na',
                'var_name','na','na','lead','na','na','na','na','MPR',
                'nobs','index','station_id','obs_lat','obs_lon',
                'level','na','fcst_value','obs_value',
                'QC','na','na']

        match_df = match_df[cols]

        # Into a list and all to strings
        mpr_data = [list( map(str,i) ) for i in match_df.values.tolist() ]

    except NameError:
        print("Can't find the input files or the variables.")
        print("Variables in this file:\t" + repr(var_list))
else:
    print("ERROR: ugrid_lfric_mpr.py -> Must specify directory of files.\n")
    sys.exit(1)

################################################################################
```

Running METplus

It is recommended to run this use case by:

Passing in StatAnalysis_fcstLFRIC_UGRID_obsASCII_PyEmbed.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/StatAnalysis_fcstLFRIC_UGRID_obsASCII_PyEmbed.conf -c /path/to/
↪user_system.conf
```

The following METplus configuration variables must be set correctly to run this example.:

- **INPUT_BASE** - Path to directory where sample data tarballs are unpacked (See Datasets section to obtain tarballs).
- **OUTPUT_BASE** - Path where METplus output will be written. This must be in a location where you have write permissions

- **MET_INSTALL_DIR** - Path to location where MET is installed locally

Example User Configuration File:

```
[dir]
INPUT_BASE = /path/to/sample/input/data
OUTPUT_BASE = /path/to/output/dir
MET_INSTALL_DIR = /path/to/met-X.Y
```

NOTE: All of these items must be found under the [dir] section.

Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

Refer to the value set for **OUTPUT_BASE** to find where the output data was generated. Output for this use case will be found in StatAnalysis_UGRID (relative to **OUTPUT_BASE**) and will contain the following file:

- dump.out

Keywords

Note:

- StatAnalysisToolUseCase
- PythonEmbeddingFileUseCase
- UnstructureGridsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1997) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/unstructured_grids-StatAnalysis_fcstLFRIC_UGRID_obsASCII_PyEmbed.png

Total running time of the script: (0 minutes 0.000 seconds)

Chapter 8

METplus Quick Search for Use Cases

Note: Use the *Keyword* after each **Use Case Type** to search for matches in the PDF version of this User's Guide.

8.1 Use Cases by MET Tool:

ASCII2NC: *ASCII2NCToolUseCase*
CyclonePlotter: *CyclonePlotterUseCase*
EnsembleStat: *EnsembleStatToolUseCase*
GenVxMask: *GenVxMaskToolUseCase*
GenEnsProd: *GenEnsProdToolUseCase*
GridStat: *GridStatToolUseCase*
GridDiag: *GridDiagToolUseCase*
IODA2NC: *IODA2NCToolUseCase*
MODE: *MODEToolUseCase*
MTD: *MTDToolUseCase*
PB2NC: *PB2NCToolUseCase*
PCPCombine: *PCPCombineToolUseCase*
Point2Grid: *Point2GridToolUseCase*
PlotDataPlane: *PlotDataPlaneToolUseCase*
PlotPointObs: *PlotPointObsToolUseCase*
PointStat: *PointStatToolUseCase*
RegridDataPlane: *RegridDataPlaneToolUseCase*
SeriesAnalysis: *SeriesAnalysisUseCase*
StatAnalysis: *StatAnalysisToolUseCase*
TCDiag: *TCDiagToolUseCase*
TCMPRPlotter: *TCMPRPlotterUseCase*

TCGen: *TCGenToolUseCase*
TCPairs: *TCPairsToolUseCase*
TCRMW: *TCRMWToolUseCase*
TCStat: *TCStatToolUseCase*

8.2 Use Cases by Application:

Air Quality and Composition: *AirQualityAndCompAppUseCase*
Climate: *ClimateAppUseCase*
Clouds: *CloudsAppUseCase*
Short Range: *ShortRangeAppUseCase*
Data Assimilation: *DataAssimilationAppUseCase*
Ensemble: *EnsembleAppUseCase*
LandSurface: *LandSurfaceAppUseCase*
Marine and Cryosphere: *MarineAndCryosphereAppUseCase*
Medium Range: *MediumRangeAppUseCase*
PBL: *PBLAppUseCase*
Precipitation: *PrecipitationAppUseCase*
Space Weather: *SpaceWeatherAppUseCase*
Subseasonal to Seasonal: *S2SAppUseCase*
Subseasonal to Seasonal: Madden-Julian Oscillation: *S2SMJOAppUseCase*
Subseasonal to Seasonal: Mid-Latitude: *S2SMidLatAppUseCase*
Tropical Cyclone and Extra-Tropical Cyclone: *TCandExtraTCAppUseCase*

8.3 Use Cases by Organization:

Developmental Testbed Center (DTC): *DTCOrgUseCase*
National Center for Atmospheric Research (NCAR): *NCAROrgUseCase*
NOAA Weather Prediction Center (WPC): *NOAAWPCOrgUseCase*
NOAA Space Weather Prediction Center (SWPC): *NOAASWPCOrgUseCase*
NOAA Environmental Modeling Center (EMC): *NOAAEMCOrgUseCase*
NOAA Global Systems Laboratory (GSL): *NOAAGSLOrgUseCase*
NOAA Hydrometeorology Testbed (HMT): *NOAAHMTOrgUseCase*
NOAA Hazardous Weather Testbed (HWT): *NOAAHWTOrgUseCase*
State University of New York-Stony Brook University (SUNY-SBU): *SBUOrgUseCase*

8.4 Use Cases by METplus Feature:

Introductory Example: *ExampleToolUseCase*
Climatology: *ClimatologyUseCase*
Custom String Looping: *CustomStringLoopingUseCase*
Diagnostics: *DiagnosticsUseCase*
Feature Relative: *FeatureRelativeUseCase*
GempakToCF: *GempakToCFToolUseCase*
GFDL Tracker: *GFDLTrackerToolUseCase*
Looping by Month or Year: *LoopByMonthFeatureUseCase*
List Expansion (using begin_end_incr syntax): *ListExpansionFeatureUseCase*
Masking for Regions of Interest: *MaskingFeatureUseCase*
METcalcpy: *METcalcpyUseCase*
METdbLoad: *METdbLoadUseCase*
METplotpy: *METplotpyUseCase*
MET_PYTHON_EXE Environment Variable: *MET_PYTHON_EXEUseCase*
Multiple Conf File Use: *MultiConfUseCase*
Observation Time Summary: *ObsTimeSummaryUseCase*
Observation Uncertainty: *ObsUncertaintyUseCase*
Python Embedding Ingest: *PyEmbedIngestToolUseCase*
Probability Generation: *ProbabilityGenerationUseCase*
Probability Verification: *ProbabilityVerificationUseCase*
Regridding in Tool: *RegriddingInToolUseCase*
Revision Series: *RevisionSeriesUseCase*
Runtime Frequency: *RuntimeFreqUseCase*
Series by Initialization: *SeriesByInitUseCase*
Series by Forecast Lead: *SeriesByLeadUseCase*
Tropical Cyclone: *TropicalCycloneUseCase*
Validation of Models or Analyses: *ValidationUseCase*
User Defined Script: *UserScriptUseCase*

8.5 Use cases by File Format:

GEMPAK: *GEMPAKFileUseCase*
GRIB: *GRIBFileUseCase*
GRIB2: *GRIB2FileUseCase*
NetCDF: *NetCDFFileUseCase*
Python Embedding: *PythonEmbeddingFileUseCase*
prepBUFR: *prepBUFRFileUseCase*

Chapter 9

METplus Configuration Glossary

<TOOL-NAME>_CLIMO_MEAN_FIELD

Specify the value for 'climo_mean.field' in the MET configuration file for <TOOL-NAME> i.e. EnsembleStat. The value set here must include the proper formatting that is expected in MET configuration file for specifying field information. Example: {name="TMP"; level="(,)";} To set the field information un-formatted, use the *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_NAME*, *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_LEVELS*, and *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_OPTIONS* variables.

Used by: Varies

<TOOL-NAME>_CLIMO_MEAN_VAR<n>_LEVELS

Specify the level of the nth field for 'climo_mean.field' in the MET configuration file for <TOOL-NAME> i.e. EnsembleStat. If any fields are set using this variable, then *<TOOL-NAME>_CLIMO_MEAN_FIELD* will be ignored. See also *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_NAME* and *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_OPTIONS*.

Used by: Varies

<TOOL-NAME>_CLIMO_MEAN_VAR<n>_NAME

Specify the name of the nth field for 'climo_mean.field' in the MET configuration file for <TOOL-NAME> i.e. EnsembleStat. If any fields are set using this variable, then *<TOOL-NAME>_CLIMO_MEAN_FIELD* will be ignored. See also *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_LEVELS* and *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_OPTIONS*.

Used by: Varies

<TOOL-NAME>_CLIMO_MEAN_VAR<n>_OPTIONS

Specify the extra options of the nth field for 'climo_mean.field' in the MET configuration file for <TOOL-NAME> i.e. EnsembleStat. If any fields are set using this variable, then *<TOOL-NAME>_CLIMO_MEAN_FIELD* will be ignored. See also *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_NAME* and *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_LEVELS*.

Used by: Varies

<TOOL-NAME>_CLIMO_STDEV_FIELD

Specify the value for 'climo_stdev.field' in the MET configuration file for <TOOL-NAME> i.e. EnsembleStat. The value set here must include the proper formatting that is expected in MET configuration file for specifying field information. Example: {name="TMP"; level="(,)";} To set the field information un-formatted, use the *<TOOL-NAME>_CLIMO_STDEV_VAR<n>_NAME*, *<TOOL-NAME>_CLIMO_STDEV_VAR<n>_LEVELS*, and *<TOOL-NAME>_CLIMO_STDEV_VAR<n>_OPTIONS* variables.

Used by: Varies

<TOOL-NAME>_CLIMO_STDEV_VAR<n>_LEVELS

Specify the level of the nth field for 'climo_stdev.field' in the MET configuration file for <TOOL-NAME> i.e. EnsembleStat. If any fields are set using this variable, then *<TOOL-NAME>_CLIMO_STDEV_FIELD* will be ignored. See also *<TOOL-NAME>_CLIMO_STDEV_VAR<n>_NAME* and *<TOOL-NAME>_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: Varies

<TOOL-NAME>_CLIMO_STDEV_VAR<n>_NAME

Specify the name of the nth field for 'climo_stdev.field' in the MET configuration file for <TOOL-NAME> i.e. EnsembleStat. If any fields are set using this variable, then *<TOOL-NAME>_CLIMO_STDEV_FIELD* will be ignored. See also *<TOOL-NAME>_CLIMO_STDEV_VAR<n>_LEVELS* and *<TOOL-NAME>_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: Varies

<TOOL-NAME>_CLIMO_STDEV_VAR<n>_OPTIONS

Specify the extra options of the nth field for 'climo_stdev.field' in the MET configuration file for <TOOL-NAME> i.e. EnsembleStat. If any fields are set using this variable, then *<TOOL-NAME>_CLIMO_STDEV_FIELD* will be ignored. See also *<TOOL-NAME>_CLIMO_STDEV_VAR<n>_NAME* and *<TOOL-NAME>_CLIMO_STDEV_VAR<n>_LEVELS*.

Used by: Varies

ADECK_FILE_PREFIX

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_ADECK_TEMPLATE*.

ADECK_TRACK_DATA_DIR

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_ADECK_INPUT_DIR*.

ALPHA_LIST

A single value or list of values used in the stat_analysis data stratification. Specifies the values of the ALPHA column in the MET .stat file to use.

Groups of values can be looped over by setting ALPHA_LIST<n> and adding ALPHA_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

AMODEL

> **Warning:** **DEPRECATED:** Please use *TC_STAT_AMODEL*.

ANLY_ASCII_REGEX_LEAD

> **Warning:** **DEPRECATED:** Please use *OBS_EXTRACT_TILES_PREFIX* instead.

ANLY_NC_TILE_REGEX

> **Warning:** **DEPRECATED:** Please use *OBS_EXTRACT_TILES_PREFIX* instead.

ANLY_TILE_PREFIX

> **Warning:** **DEPRECATED:** Please use *OBS_EXTRACT_TILES_PREFIX* instead.

ANLY_TILE_REGEX

Warning: **DEPRECATED:** No longer used. The regular expression for the analysis input file. The file is in GRIBv2 format.

ASCII2NC_CONFIG_FILE
Path to optional configuration file read by ascii2nc. To utilize a configuration file, set this to {PARM_BASE}/met_config/Ascii2NcConfig_wrapped. If unset, no config file will be used.

Used by: ASCII2NC

ASCII2NC_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: ASCII2NC

ASCII2NC_FILE_WINDOW_BEGIN
Used to control the lower bound of the window around the valid time to determine if an ASCII2NC input file should be used for processing. Overrides *OBS_FILE_WINDOW_BEGIN*. See 'Use Windows to Find Valid Files' section for more information.

Used by: ASCII2NC

ASCII2NC_FILE_WINDOW_END
Used to control the upper bound of the window around the valid time to determine if an ASCII2NC input file should be used for processing. Overrides *OBS_FILE_WINDOW_END*. See 'Use Windows to Find Valid Files' section for more information.

Used by: ASCII2NC

ASCII2NC_INPUT_DIR
Directory containing input data to ASCII2NC. This variable is optional because you can specify the full path to the input files using *ASCII2NC_INPUT_TEMPLATE*.

Used by: ASCII2NC

ASCII2NC_INPUT_FORMAT
Optional string to specify the format of the input data. Valid options are "met_point", "little_r", "surfrad", "wwsis", "aeronet", "aeronetv2", or "aeronetv3."

Used by: ASCII2NC

ASCII2NC_INPUT_TEMPLATE
Filename template of the input file used by ASCII2NC. See also *ASCII2NC_INPUT_DIR*.

Used by: ASCII2NC

ASCII2NC_MASK_GRID
Named grid or a data file defining the grid for filtering the point observations spatially (optional).

Used by: ASCII2NC

ASCII2NC_MASK_POLY
A polyline file, the output of gen_vx_mask, or a gridded data file with field information for filtering the point observations spatially (optional).

Used by: ASCII2NC

ASCII2NC_MASK_SID
A station ID masking file or a comma-separated list of station ID's for filtering the point observations spatially (optional).

Used by: ASCII2NC

ASCII2NC_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: ASCII2NC_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: ASCII2NC

ASCII2NC_OUTPUT_DIR
Directory to write output data generated by ASCII2NC. This variable is optional because you can specify the full path to the output files using *ASCII2NC_OUTPUT_TEMPLATE*.

Used by: ASCII2NC

ASCII2NC_OUTPUT_TEMPLATE
Filename template of the output file generated by ASCII2NC. See also *ASCII2NC_OUTPUT_DIR*.

Used by: ASCII2NC

ASCII2NC_SKIP_IF_OUTPUT_EXISTS
If True, do not run ASCII2NC if output file already exists. Set to False to overwrite files.

Used by: ASCII2NC

ASCII2NC_TIME_SUMMARY_BEG
Read by the ASCII2NC configuration file if specified by *ASCII2NC_CONFIG_FILE*. See the MET User's Guide section regarding ASCII2NC configuration files for more information.

Used by: ASCII2NC

ASCII2NC_TIME_SUMMARY_END
Read by the ASCII2NC configuration file if specified by *ASCII2NC_CONFIG_FILE*. See the MET User's Guide section regarding ASCII2NC configuration files for more information.

Used by: ASCII2NC

ASCII2NC_TIME_SUMMARY_FLAG
Boolean value to turn on/off time summarization. Read by the ASCII2NC configuration file if specified by *ASCII2NC_CONFIG_FILE*. See the MET User's Guide section regarding ASCII2NC configuration for more information.

Used by: ASCII2NC

ASCII2NC_TIME_SUMMARY_GRIB_CODES
Read by the ASCII2NC configuration file if specified by *ASCII2NC_CONFIG_FILE*. See the MET User's Guide section regarding ASCII2NC configuration files for more information.

Used by: ASCII2NC

ASCII2NC_TIME_SUMMARY_RAW_DATA
Read by the ASCII2NC configuration file if specified by *ASCII2NC_CONFIG_FILE*. See the MET User's Guide section regarding ASCII2NC configuration files for more information.

Used by: ASCII2NC

ASCII2NC_TIME_SUMMARY_STEP
Read by the ASCII2NC configuration file if specified by *ASCII2NC_CONFIG_FILE*. See the MET User's Guide section regarding ASCII2NC configuration files for more information.

Used by: ASCII2NC

ASCII2NC_TIME_SUMMARY_TYPES
Read by the ASCII2NC configuration file if specified by *ASCII2NC_CONFIG_FILE*. See the MET User's Guide section regarding ASCII2NC configuration files for more information.

Used by: ASCII2NC

ASCII2NC_TIME_SUMMARY_VALID_FREQ
Read by the ASCII2NC configuration file if specified by *ASCII2NC_CONFIG_FILE*. See the MET User's Guide section regarding ASCII2NC configuration files for more information.

Used by: ASCII2NC

ASCII2NC_TIME_SUMMARY_VALID_THRESH
Read by the ASCII2NC configuration file if specified by *ASCII2NC_CONFIG_FILE*. See the MET User's Guide section regarding ASCII2NC configuration files for more information.

Used by: ASCII2NC

ASCII2NC_TIME_SUMMARY_VAR_NAMES

Read by the ASCII2NC configuration file if specified by *ASCII2NC_CONFIG_FILE*. See the MET User's Guide section regarding ASCII2NC configuration files for more information.

Used by: ASCII2NC

ASCII2NC_TIME_SUMMARY_WIDTH

Read by the ASCII2NC configuration file if specified by *ASCII2NC_CONFIG_FILE*. See the MET User's Guide section regarding ASCII2NC configuration files for more information.

Used by: ASCII2NC

ASCII2NC_WINDOW_BEGIN

Passed to the ASCII2NC MET config file to determine the range of data within a file that should be used for processing. Units are seconds. If the variable is not set, ASCII2NC will use *OBS_WINDOW_BEGIN*.

Used by: ASCII2NC

ASCII2NC_WINDOW_END

Passed to the ASCII2NC MET config file to determine the range of data within a file that should be used for processing. Units are seconds. If the variable is not set, ASCII2NC will use *OBS_WINDOW_END*.

Used by: ASCII2NC

BACKGROUND_MAP

Warning: **DEPRECATED:** Please use *SERIES_ANALYSIS_BACKGROUND_MAP* instead.

BASIN

Warning: **DEPRECATED:** Please use *TC_PAIRS_BASIN* or *TC_STAT_BASIN*.

BDECK_FILE_PREFIX

Warning: **DEPRECATED:** Please use *TC_PAIRS_BDECK_TEMPLATE*.

BDECK_TRACK_DATA_DIR

Warning: **DEPRECATED:** Please use *TC_PAIRS_BDECK_INPUT_DIR*.

BEG_TIME

Warning: **DEPRECATED:** Please use *INIT_BEG* or *VALID_BEG* instead.

BMODEL

Warning: **DEPRECATED:** Please use *TC_STAT_BMODEL*.

BOTH_SERIES_ANALYSIS_INPUT_DIR
Specify the directory to read forecast and observation input from the same file in SeriesAnalysis. See also *BOTH_SERIES_ANALYSIS_INPUT_TEMPLATE*

Used by: SeriesAnalysis

BOTH_SERIES_ANALYSIS_INPUT_FILE_LIST
Specifies an explicit path to a file list file to pass into series_analysis with the -both argument. If set, *BOTH_SERIES_ANALYSIS_INPUT_TEMPLATE* and *BOTH_SERIES_ANALYSIS_INPUT_DIR* are ignored. See also *FCST_SERIES_ANALYSIS_INPUT_FILE_LIST* and *OBS_SERIES_ANALYSIS_INPUT_FILE_LIST*.

Used by: SeriesAnalysis

BOTH_SERIES_ANALYSIS_INPUT_TEMPLATE
Template to find forecast and observation input from the same file in SeriesAnalysis. See also *BOTH_SERIES_ANALYSIS_INPUT_DIR*

Used by: SeriesAnalysis

BOTH_VAR<n>_LEVELS
Define the levels for the <n>th forecast and observation variables to be used in the analysis where

<n> is an integer >= 1. See *FCST_VAR<n>_LEVELS*, *OBS_VAR<n>_LEVELS*, or *Field Info* (page 58) for more information.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

BOTH_VAR<n>_NAME

Define the name for the <n>th forecast and observation variables to be used in the analysis where <n> is an integer >= 1. See *FCST_VAR<n>_NAME*, *OBS_VAR<n>_NAME*, or *Field Info* (page 58) for more information.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

BOTH_VAR<n>_OPTIONS

Define the extra options for the <n>th forecast and observation variables to be used in the analysis where <n> is an integer >= 1. See *FCST_VAR<n>_OPTIONS*, *OBS_VAR<n>_OPTIONS*, or *Field Info* (page 58) for more information.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

BOTH_VAR<n>_THRESH

Define the threshold list for the <n>th forecast and observation variables to be used in the analysis where <n> is an integer >= 1. See *FCST_VAR<n>_THRESH*, *OBS_VAR<n>_THRESH*, or *Field Info* (page 58) for more information.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

CLIMO_GRID_STAT_INPUT_DIR

Warning: **DEPRECATED:** Please use *GRID_STAT_CLIMO_MEAN_FILE_NAME*.

CLIMO_GRID_STAT_INPUT_TEMPLATE

Warning: **DEPRECATED:** Please use *GRID_STAT_CLIMO_MEAN_FILE_NAME*.

CLIMO_POINT_STAT_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *POINT_STAT_CLIMO_MEAN_FILE_NAME*.

CLIMO_POINT_STAT_INPUT_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *POINT_STAT_CLIMO_MEAN_FILE_NAME*.

CLOCK_TIME
Automatically set by METplus with the time that the run was started. Setting this variable has no effect as it will be overwritten. Can be used for reference in metplus_final.conf or used with other config variables.

Used by: All

CONFIG_DIR
Directory containing config files relevant to MET tools.

Used by: EnsembleStat, GridStat, MODE, StatAnalysis

CONFIG_FILE

> **Warning:** **DEPRECATED:** Please use *TCMPR_PLOTTER_CONFIG_FILE*.

Used by: TCMPRPlotter

CONVERT
Path to the ImageMagick convert executable.

Used by: PlotDataPlane

CONVERT_EXE

> **Warning:** **DEPRECATED:** Please use *CONVERT*.

COV_THRESH

> **Warning:** **DEPRECATED:** Please use *COV_THRESH_LIST* instead.

COV_THRESH_LIST
Specify the values of the COV_THRESH column in the MET .stat file to use;

Used by: StatAnalysis

CURRENT_FCST_LEVEL
Generated by METplus in wrappers that loop over forecast names/levels to keep track of the current forecast level that is being processed. It can be referenced in the [GRID_STAT/MODE/MTD]_OUTPUT_PREFIX to set the output file names. This should not be set by a user!

Used by: GridStat, MODE, MTD

CURRENT_FCST_NAME
Generated by METplus in wrappers that loop over forecast names/levels to keep track of the current forecast name that is being processed. It can be referenced in the [GRID_STAT/MODE/MTD]_OUTPUT_PREFIX to set the output file names. This should not be set by a user!

Used by: GridStat, MODE, MTD

CURRENT_OBS_LEVEL
Generated by METplus in wrappers that loop over observation names/levels to keep track of the current observation level that is being processed. It can be referenced in the [GRID_STAT/MODE/MTD]_OUTPUT_PREFIX to set the output file names. This should not be set by a user!

Used by: GridStat, MODE, MTD

CURRENT_OBS_NAME
Generated by METplus in wrappers that loop over observation names/levels to keep track of the current observation name that is being processed. It can be referenced in the [GRID_STAT/MODE/MTD]_OUTPUT_PREFIX to set the output file names. This should not be set by a user!

Used by: GridStat, MODE, MTD

CUSTOM_INGEST_<n>_OUTPUT_DIR

> **Warning:** **DEPRECATED:** Please use *PY_EMBED_INGEST_<n>_OUTPUT_DIR*.

CUSTOM_INGEST_<n>_OUTPUT_GRID

> **Warning:** **DEPRECATED:** Please use *PY_EMBED_INGEST_<n>_OUTPUT_GRID*.

CUSTOM_INGEST_<n>_OUTPUT_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *PY_EMBED_INGEST_<n>_OUTPUT_TEMPLATE*.

CUSTOM_INGEST_<n>_SCRIPT

> **Warning:** **DEPRECATED:** Please use *PY_EMBED_INGEST_<n>_SCRIPT*.

CUSTOM_INGEST_<n>_TYPE

> **Warning:** **DEPRECATED:** Please use *PY_EMBED_INGEST_<n>_TYPE*.

CUSTOM_LOOP_LIST

List of strings that are used to run each item in the *PROCESS_LIST* multiple times for each run time to allow the tool to be run with different configurations. The filename template tag {custom?fmt=%s} can be used throughout the METplus configuration file. For example, the text can be used to supply different configuration files (if the MET tool uses them) and output filenames/directories. If you have two configuration files, SeriesAnalysisConfig_one and SeriesAnalysisConfig_two, you can set:

```
[config]
CUSTOM_LOOP_LIST = one, two
SERIES_ANALYSIS_CONFIG_FILE = {CONFIG_DIR}/SeriesAnalysisConfig_{custom?fmt=%s}

[dir]
SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/{custom?fmt=%s}
```

With this configuration, SeriesAnalysis will be called twice. The first run will use SeriesAnalysisConfig_one and write output to {OUTPUT_BASE}/one. The second run will use SeriesAnalysisConfig_two and write output to {OUTPUT_BASE}/two.

If unset or left blank, the wrapper will run once per run time. There are also wrapper-specific configuration variables to define a custom string loop list for a single wrapper, i.e. *SERIES_ANALYSIS_CUSTOM_LOOP_LIST* and *PCP_COMBINE_CUSTOM_LOOP_LIST*.

Used by: Many

CUT

Path to the Linux cut executable.

Used by: PB2NC, PointStat

CUT_EXE

Warning: **DEPRECATED:** Please use *CUT*.

CYCLONE

Warning: **DEPRECATED:** Please use *TC_PAIRS_CYCLONE* or *TC_STAT_CYCLONE*.

CYCLONE_CIRCLE_MARKER_SIZE

Warning: **DEPRECATED:** Please use *CYCLONE_PLOTTER_CIRCLE_MARKER_SIZE*.

CYCLONE_CROSS_MARKER_SIZE

Warning: **DEPRECATED:** Please use *CYCLONE_PLOTTER_CROSS_MARKER_SIZE*.

CYCLONE_GENERATE_TRACK_ASCII

Warning: **DEPRECATED:** Please use *CYCLONE_PLOTTER_GENERATE_TRACK_ASCII* instead.

CYCLONE_INIT_DATE

Warning: **DEPRECATED:** Please use *CYCLONE_PLOTTER_INIT_DATE* instead.

CYCLONE_INIT_HR
Initialization hour for the cyclone forecasts in HH format.

Used by: CyclonePlotter

CYCLONE_INPUT_DIR
Input directory for the cyclone plotter. This should be the output directory for the MET TC-Pairs utility

Used by: CyclonePlotter

CYCLONE_MODEL
Define the model being used for the tropical cyclone forecasts.

Used by: CyclonePlotter

CYCLONE_OUT_DIR
Specify the directory where the output from the cyclone plotter should go.

Used by: CyclonePlotter

CYCLONE_PLOT_TITLE

> **Warning:** **DEPRECATED:** Please use *CYCLONE_PLOTTER_PLOT_TITLE*.

CYCLONE_PLOTTER_ADD_WATERMARK
If set to True, add a watermark with the current time to the image generated by CyclonePlotter.

Used by: CyclonePlotter

CYCLONE_PLOTTER_ANNOTATION_FONT_SIZE
Set the annotation font size for CyclonePlotter output.

Used by: CyclonePlotter

CYCLONE_PLOTTER_CIRCLE_MARKER_SIZE
Control the size of the circle marker in the cyclone plotter.

Used by: CyclonePlotter

CYCLONE_PLOTTER_CROSS_MARKER_SIZE
Control the size of the cross marker in the cyclone plotter.

Used by: CyclonePlotter

CYCLONE_PLOTTER_EAST_LON
Set the east longitude boundary for CyclonePlotter. Only used if *CYCLONE_PLOTTER_GLOBAL_PLOT* is False.

Used by: CyclonePlotter

CYCLONE_PLOTTER_GENERATE_TRACK_ASCII
Specify whether or not to produce an ASCII file containing all of the tracks in the plot. Acceptable values: true/false

Used by: CyclonePlotter

CYCLONE_PLOTTER_GLOBAL_PLOT
Set to True to plot entire global extent in CyclonePlotter or set to False to generate a plot of a defined region of the world, then define lons and lats with *CYCLONE_PLOTTER_WEST_LON*, *CYCLONE_PLOTTER_EAST_LON*, *CYCLONE_PLOTTER_SOUTH_LAT*, and *CYCLONE_PLOTTER_NORTH_LAT*.

Used by: CyclonePlotter

CYCLONE_PLOTTER_INIT_DATE
Initialization date for the cyclone forecasts in YYYYMMDD format.

Used by: CyclonePlotter

CYCLONE_PLOTTER_INIT_HR

> **Warning:** **DEPRECATED:** Please use *CYCLONE_PLOTTER_INIT_DATE* instead.

CYCLONE_PLOTTER_INPUT_DIR
The directory containing the input data to be plotted.

Used by: CyclonePlotter

CYCLONE_PLOTTER_MODEL
Model used in CyclonePlotter.

Used by: CyclonePlotter

CYCLONE_PLOTTER_NORTH_LAT
Set the north latitude boundary for CyclonePlotter. Only used if *CYCLONE_PLOTTER_GLOBAL_PLOT* is False.

Used by: CyclonePlotter

CYCLONE_PLOTTER_OUTPUT_DIR
Directory for saving files generated by CyclonePlotter.

Used by: CyclonePlotter

CYCLONE_PLOTTER_PLOT_TITLE
Title string for the cyclone plotter.

Used by: CyclonePlotter

CYCLONE_PLOTTER_RESOLUTION_DPI
Set the resolution for CyclonePlotter output.

Used by: CyclonePlotter

CYCLONE_PLOTTER_SOUTH_LAT
Set the south latitude boundary for CyclonePlotter. Only used if *CYCLONE_PLOTTER_GLOBAL_PLOT* is False.

Used by: CyclonePlotter

CYCLONE_PLOTTER_WEST_LON
Set the west longitude boundary for CyclonePlotter. Only used if *CYCLONE_PLOTTER_GLOBAL_PLOT* is False.

Used by: CyclonePlotter

DATE_TYPE
In StatAnalysis, this specifies the way to treat the date information, where valid options are VALID and INIT.

Used by: StatAnalysis

DEMO_YR

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_DEMO_YR* instead.

DEP_VARS

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_DEP_VARS* instead.

DESC
Specify the value for 'desc' in the MET configuration file for the MET tool being used

Used by: GridStat, PointStat, EnsembleStat, GridDiag, MODE, MTD, SeriesAnalysis, TCGen, TCPairs, TCStat

DESC_LIST
A single value or list of values used in the stat_analysis data stratification. Specifies the values of the DESC column in the MET .stat file to use.

Groups of values can be looped over by setting DESC_LIST<n> and adding DESC_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

DLAND_FILE

Warning: **DEPRECATED:** Please use *TC_PAIRS_DLAND_FILE*.

DLAT

Warning: **DEPRECATED:** Please use *EXTRACT_TILES_DLAT* instead.

DLON

Warning: **DEPRECATED:** Please use *EXTRACT_TILES_DLON* instead.

DO_NOT_RUN_EXE

True/False. If True, applications will not run and will only output command that would have been called.

Used by: All

END_DATE

Warning: **DEPRECATED:** Please use *INIT_END* or *VALID_END* instead.

END_HOUR

Warning: **DEPRECATED:** Ending hour for analysis with format HH.

END_TIME

Warning: **DEPRECATED:** Ending date string for analysis with format YYYYMMDD.

ENS_ENSEMBLE_STAT_INPUT_DATATYPE

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_INPUT_DATATYPE* in *GenEnsProd* (page 119) instead.

ENS_VAR<n>_LEVELS

Define the levels for the <n>th ensemble variable to be used in the analysis where <n> is an integer >= 1. The value can be a single item or a comma separated list of items. You can define NetCDF

levels, such as (0,*,*), but you will need to surround these values with quotation marks so that the commas in the item are not interpreted as an item delimeter. Some examples:

ENS_VAR1_LEVELS = A06, P500
ENS_VAR2_LEVELS ="(0,*,*)", "(1,*,*)"

There can be <n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.:

ENS_VAR1_LEVELS
ENS_VAR2_LEVELS
...
ENS_VAR<n>_LEVELS

See *Field Info* (page 58) for more information.

Used by: EnsembleStat

ENS_VAR<n>_NAME

Define the name for the <n>th ensemble variable to be used in the analysis where <n> is an integer >= 1. There can be <n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.:

ENS_VAR1_NAME
ENS_VAR2_NAME
...
ENS_VAR<n>_NAME

See *Field Info* (page 58) for more information.

Used by: EnsembleStat

ENS_VAR<n>_OPTIONS

Define the options for the <n>th ensemble variable to be used in the analysis where <n> is an integer >= 1. These addition options will be applied to every name/level/threshold combination

for VAR<n>. There can be <n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.:

ENS_VAR1_OPTIONS
ENS_VAR2_OPTIONS
...
ENS_VAR<n>_OPTION

See *Field Info* (page 58) for more information.

Used by: EnsembleStat

ENS_VAR<n>_THRESH

Define the threshold(s) for the <n>th ensemble variable to be used in the analysis where <n> is an integer >= 1. The value can be a single item or a comma separated list of items that must start with a comparison operator (>,>=,==,!=,<,<=,gt,ge,eq,ne,lt,le). There can be <n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.:

ENS_VAR1_THRESH
ENS_VAR2_THRESH
...
ENS_VAR<n>_THRESH

See *Field Info* (page 58) for more information.

Used by: EnsembleStat

ENSEMBLE_STAT_CENSOR_THRESH

Specify the value for 'censor_thresh' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CENSOR_VAL

Specify the value for 'censor_val' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CI_ALPHA
Specify the value for 'ci_alpha' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_CDF_BINS
Specify the value for 'climo_cdf.cdf_bins' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_CDF_CDF_BINS
See *ENSEMBLE_STAT_CLIMO_CDF_BINS*

ENSEMBLE_STAT_CLIMO_CDF_CENTER_BINS
Specify the value for 'climo_cdf.center_bins' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_CDF_DIRECT_PROB
Specify the value for 'climo_cdf.direct_prob' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_CDF_WRITE_BINS
Specify the value for 'climo_cdf.write_bins' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_DAY_INTERVAL
Specify the value for 'climo_mean.day_interval' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_FIELD
See: *<TOOL-NAME>_CLIMO_MEAN_FIELD*

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_FILE_NAME
Specify the value for 'climo_mean.file_name' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_HOUR_INTERVAL
Specify the value for 'climo_mean.hour_interval' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *ENSEMBLE_STAT_CLIMO_MEAN_FILE_NAME*.

ENSEMBLE_STAT_CLIMO_MEAN_INPUT_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *ENSEMBLE_STAT_CLIMO_MEAN_FILE_NAME*.

ENSEMBLE_STAT_CLIMO_MEAN_MATCH_MONTH
Specify the value for 'climo_mean.match_month' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_REGRID_METHOD
Specify the value for 'climo_mean.regrid.method' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_REGRID_SHAPE
Specify the value for 'climo_mean.regrid.shape' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_REGRID_VLD_THRESH
Specify the value for 'climo_mean.regrid.vld_thresh' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_REGRID_WIDTH
Specify the value for 'climo_mean.regrid.width' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_TIME_INTERP_METHOD
Specify the value for 'climo_mean.time_interp_method' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_USE_FCST
If set to True, use the field array from the fcst dictionary for the climo_mean fields for EnsembleStat. Sets "climo_mean = fcst;" in the wrapped MET config file. Only used if *ENSEMBLE_STAT_CLIMO_MEAN_FIELD* is unset. See also *ENSEMBLE_STAT_CLIMO_MEAN_USE_OBS*.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_USE_OBS
If set to True, use the field array from the obs dictionary for the climo_mean fields for EnsembleStat. Sets "climo_mean = obs;" in the wrapped MET config file. Only used if *ENSEMBLE_STAT_CLIMO_MEAN_FIELD* is unset. See also *ENSEMBLE_STAT_CLIMO_MEAN_USE_FCST*.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_VAR<n>_LEVELS
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_LEVELS*

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_VAR<n>_NAME
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_NAME*

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_MEAN_VAR<n>_OPTIONS
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_OPTIONS*

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_DAY_INTERVAL
Specify the value for 'climo_stdev.day_interval' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_FIELD
Specify the value for 'climo_stdev.field' in the MET configuration file for EnsembleStat. The value set here must include the proper formatting that is expected in MET configuration file for specifying field information. Example: {name="TMP"; level="(,)";} To set the field information un-formatted, use the *ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_NAME*, *ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_LEVELS*, and *ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_OPTIONS* variables.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_FILE_NAME
Specify the value for 'climo_stdev.file_name' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_HOUR_INTERVAL
Specify the value for 'climo_stdev.hour_interval' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *ENSEMBLE_STAT_CLIMO_STDEV_FILE_NAME*.

ENSEMBLE_STAT_CLIMO_STDEV_INPUT_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *ENSEMBLE_STAT_CLIMO_STDEV_FILE_NAME*.

ENSEMBLE_STAT_CLIMO_STDEV_MATCH_MONTH
Specify the value for 'climo_stdev.match_month' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_REGRID_METHOD
Specify the value for 'climo_stdev.regrid.method' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_REGRID_SHAPE
Specify the value for 'climo_stdev.regrid.shape' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_REGRID_VLD_THRESH
Specify the value for 'climo_stdev.regrid.vld_thresh' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_REGRID_WIDTH
Specify the value for 'climo_stdev.regrid.width' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_TIME_INTERP_METHOD
Specify the value for 'climo_stdev.time_interp_method' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_USE_FCST

If set to True, use the field array from the fcst dictionary for the climo_stdev fields for EnsembleStat. Sets "climo_stdev = fcst;" in the wrapped MET config file. Only used if *ENSEMBLE_STAT_CLIMO_STDEV_FIELD* is unset. See also *ENSEMBLE_STAT_CLIMO_STDEV_USE_OBS*.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_USE_OBS

If set to True, use the field array from the obs dictionary for the climo_stdev fields for EnsembleStat. Sets "climo_stdev = obs;" in the wrapped MET config file. Only used if *ENSEMBLE_STAT_CLIMO_STDEV_FIELD* is unset. See also *ENSEMBLE_STAT_CLIMO_STDEV_USE_FCST*.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_LEVELS

Specify the level of the nth field for 'climo_stdev.field' in the MET configuration file for EnsembleStat. If any fields are set using this variable, then *ENSEMBLE_STAT_CLIMO_STDEV_FIELD* will be ignored. See also *ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_NAME* and *ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_NAME

Specify the name of the nth field for 'climo_stdev.field' in the MET configuration file for EnsembleStat. If any fields are set using this variable, then *ENSEMBLE_STAT_CLIMO_STDEV_FIELD* will be ignored. See also *ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_LEVELS* and *ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: EnsembleStat

ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_OPTIONS

Specify the extra options of the nth field for 'climo_stdev.field' in the MET configuration file for EnsembleStat. If any fields are set using this variable, then *ENSEMBLE_STAT_CLIMO_STDEV_FIELD* will be ignored. See also *ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_NAME* and *ENSEMBLE_STAT_CLIMO_STDEV_VAR<n>_LEVELS*.

Used by: EnsembleStat

ENSEMBLE_STAT_CONFIG

> **Warning: DEPRECATED:** Please use *ENSEMBLE_STAT_CONFIG_FILE* instead.

ENSEMBLE_STAT_CONFIG_FILE
Path to configuration file read by ensemble_stat. If unset, parm/met_config/EnsembleStatConfig_wrapped will be used.

Used by: EnsembleStat

ENSEMBLE_STAT_CONTROL_ID
Specify the value for 'control_id' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_CTRL_INPUT_DIR
Input directory for optional control file to use with EnsembleStat. See also *ENSEMBLE_STAT_CTRL_INPUT_TEMPLATE*.

Used by: EnsembleStat

ENSEMBLE_STAT_CTRL_INPUT_TEMPLATE
Template used to specify an optional control filename for EnsembleStat. Note that if a control member file is found in the ensemble file list, it will automatically be removed by the wrapper to prevent an error in the MET tool. This may require adjusting the value for *ENSEMBLE_STAT_N_MEMBERS* and/or *ENSEMBLE_STAT_VLD_THRESH*.

Used by: EnsembleStat

ENSEMBLE_STAT_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: EnsembleStat

ENSEMBLE_STAT_DESC
Specify the value for 'desc' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_DUPLICATE_FLAG
Specify the value for 'duplicate_flag' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ECLV_POINTS
Specify the value for 'eclv_points' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENS_MEAN_INPUT_DIR
Input directory for the optional -ens_mean file to use with the MET tool ensemble_stat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENS_MEAN_INPUT_TEMPLATE
Template used to specify the optional -ens_mean file for the MET tool ensemble_stat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENS_MEMBER_IDS
Specify the value for 'ens_member_ids' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENS_OBS_THRESH

Warning: **DEPRECATED:** Please use *ENSEMBLE_STAT_OBS_THRESH*.

ENSEMBLE_STAT_ENS_PHIST_BIN_SIZE
Specify the value for 'ens_phist_bin_size' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENS_SSVAR_BIN_SIZE

Specify the value for 'ens_ssvar_bin_size' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENS_THRESH

Threshold for the ratio of the number of valid ensemble fields to the total number of expected ensemble members. This value is passed into the ensemble_stat config file to make sure the percentage of files that are valid meets the expectation.

Used by: EnsembleStat

ENSEMBLE_STAT_ENS_VLD_THRESH

Warning: **DEPRECATED:** Please use *ENSEMBLE_STAT_VLD_THRESH* instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_LATLON

Warning: **DEPRECATED:** Please use *ENSEMBLE_STAT_NC_ORANK_FLAG_LATLON* or *GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_ENSEMBLE_FLAG_MAX* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN

Warning: **DEPRECATED:** Please use *ENSEMBLE_STAT_NC_ORANK_FLAG_MEAN* or *GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_ENSEMBLE_FLAG_MIN* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_ENSEMBLE_FLAG_MINUS* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_ENSEMBLE_FLAG_NEP* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_ENSEMBLE_FLAG_NMEP* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_ENSEMBLE_FLAG_PLUS* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_ENSEMBLE_FLAG_RANGE* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK

> **Warning:** **DEPRECATED:** Please use *ENSEMBLE_STAT_NC_ORANK_FLAG_RANK* instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV

> **Warning:** **DEPRECATED:** Please use *GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT

> **Warning:** **DEPRECATED:** Please use *ENSEMBLE_STAT_NC_ORANK_FLAG_VLD_COUNT* or *GEN_ENS_PROD_ENSEMBLE_FLAG_VLD_COUNT* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_ENSEMBLE_FLAG_WEIGHT

> **Warning:** **DEPRECATED:** Please use *ENSEMBLE_STAT_NC_ORANK_FLAG_WEIGHT* instead.

ENSEMBLE_STAT_GRID_VX

> **Warning:** **DEPRECATED:** Please use *ENSEMBLE_STAT_REGRID_TO_GRID*.

ENSEMBLE_STAT_GRID_WEIGHT_FLAG

Specify the value for 'grid_weight_flag' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_INTERP_FIELD

Specify the value for 'interp.field' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_INTERP_METHOD

Specify the value for 'interp.type.method' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_INTERP_SHAPE
Specify the value for 'interp.shape' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_INTERP_VLD_THRESH
Specify the value for 'interp.vld_thresh' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_INTERP_WIDTH
Specify the value for 'interp.type.width' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_MASK_GRID
Specify the value for 'mask.grid' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_MASK_POLY
Set the mask.poly entry in the EnsembleStat MET config file.

Used by: EnsembleStat

ENSEMBLE_STAT_MESSAGE_TYPE
Set the message_type option in the EnsembleStat MET config file.

Used by: EnsembleStat

ENSEMBLE_STAT_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: ENSEMBLE_STAT_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: EnsembleStat

ENSEMBLE_STAT_MET_OBS_ERR_TABLE

Used by: EnsembleStat

ENSEMBLE_STAT_MET_OBS_ERROR_TABLE

Warning: **DEPRECATED:** Please use *ENSEMBLE_STAT_MET_OBS_ERR_TABLE* instead.

ENSEMBLE_STAT_N_MEMBERS

Expected number of ensemble members found. This should correspond to the number of items in *FCST_ENSEMBLE_STAT_INPUT_TEMPLATE*. If this number differs from the number of files are found for a given run, then ensemble_stat will not run for that time.

Used by: EnsembleStat

ENSEMBLE_STAT_NBRHD_PROB_SHAPE

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_NBRHD_PROB_SHAPE* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_NBRHD_PROB_VLD_THRESH

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_NBRHD_PROB_VLD_THRESH* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_NBRHD_PROB_WIDTH

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_NBRHD_PROB_WIDTH* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_NC_ORANK_FLAG_LATLON

Specify the value for 'nc_orank_flag.latlon' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NC_ORANK_FLAG_MEAN
Specify the value for 'nc_orank_flag.mean' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NC_ORANK_FLAG_PIT
Specify the value for 'nc_orank_flag.pit' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NC_ORANK_FLAG_RANK
Specify the value for 'nc_orank_flag.rank' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NC_ORANK_FLAG_RAW
Specify the value for 'nc_orank_flag.raw' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NC_ORANK_FLAG_VLD_COUNT
Specify the value for 'nc_orank_flag.vld_count' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NC_ORANK_FLAG_WEIGHT
Specify the value for 'nc_orank_flag.weight' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_DX

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_DX* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_RADIUS

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_RADIUS* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_NMEP_SMOOTH_METHOD

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_NMEP_SMOOTH_METHOD* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_NMEP_SMOOTH_SHAPE

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_NMEP_SMOOTH_SHAPE* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_NMEP_SMOOTH_VLD_THRESH

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_NMEP_SMOOTH_VLD_THRESH* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_NMEP_SMOOTH_WIDTH

Warning: **DEPRECATED:** Please use *GEN_ENS_PROD_NMEP_SMOOTH_WIDTH* in *GenEnsProd* (page 119) instead.

ENSEMBLE_STAT_OBS_ERROR_FLAG
Specify the value for 'obs_error.flag' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OBS_QUALITY_EXC
Specify the value for 'obs_quality_exc' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OBS_QUALITY_INC
Specify the value for 'obs_quality_inc' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OBS_THRESH
Sets the obs_thresh value in the ensemble_stat MET config file.

Used by: EnsembleStat

ENSEMBLE_STAT_OUT_DIR

> **Warning:** **DEPRECATED:** Please use *ENSEMBLE_STAT_OUTPUT_DIR* instead.

ENSEMBLE_STAT_OUTPUT_DIR
Specify the output directory where files from the MET ensemble_stat tool are written.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_ECLV
Specify the value for 'output_flag.eclv' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_ECNT
Specify the value for 'output_flag.ecnt' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_ORANK
Specify the value for 'output_flag.orank' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_PCT
Specify the value for 'output_flag.pct' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_PHIST
Specify the value for 'output_flag.phist' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_PJC
Specify the value for 'output_flag.pjc' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_PRC
Specify the value for 'output_flag.prc' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_PSTD
Specify the value for 'output_flag.pstd' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_RELP
Specify the value for 'output_flag.relp' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_RHIST
Specify the value for 'output_flag.rhist' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_RPS
Specify the value for 'output_flag.rps' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_FLAG_SSVAR
Specify the value for 'output_flag.ssvar' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_PREFIX
String to pass to the MET config file to prepend text to the output filenames.

Used by: EnsembleStat

ENSEMBLE_STAT_OUTPUT_TEMPLATE
Sets the subdirectories below *ENSEMBLE_STAT_OUTPUT_DIR* using a template to allow run time information. If *LOOP_BY* = VALID, default value is valid time YYYYMMDDHHMM/ensemble_stat. If *LOOP_BY* = INIT, default value is init time YYYYMMDDHHMM/ensemble_stat.

Used by: EnsembleStat

ENSEMBLE_STAT_PROB_CAT_THRESH
Specify the value for 'prob_cat_thresh' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_PROB_PCT_THRESH
Specify the value for 'prob_pct_thresh' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_REGRID_CENSOR_THRESH
Specify the value for 'regrid.censor_thresh' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_REGRID_CENSOR_VAL
Specify the value for 'regrid.censor_val' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_REGRID_CONVERT
Specify the value for 'regrid.convert' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_REGRID_METHOD
Specify the value for 'regrid.method' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_REGRID_SHAPE
Specify the value for 'regrid.shape' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_REGRID_TO_GRID
Used to set the regrid dictionary item 'to_grid' in the MET EnsembleStat config file. See the MET User's Guide for more information.

Used by: EnsembleStat

ENSEMBLE_STAT_REGRID_VLD_THRESH
Specify the value for 'regrid.vld_thresh' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_REGRID_WIDTH
Specify the value for 'regrid.width' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_SKIP_CONST
Specify the value for 'skip_const' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: EnsembleStat

ENSEMBLE_STAT_VERIFICATION_MASK_TEMPLATE
Template used to specify the verification mask filename for the MET tool ensemble_stat. Now supports a list of filenames.

Used by: EnsembleStat

ENSEMBLE_STAT_VLD_THRESH
Threshold for the ratio of the number of valid data values to the total number of expected ensemble members. This value is passed into the ensemble_stat config file to make sure the percentage of files that are valid meets the expectation.

Used by: EnsembleStat

EXAMPLE_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: Example

EXAMPLE_INPUT_DIR
Directory containing fake input data for Example wrapper. This variable is optional because you can specify the full path to the input files using *EXAMPLE_INPUT_TEMPLATE*.

Used by: Example

EXAMPLE_INPUT_TEMPLATE
Filename template of the fake input files used by Example wrapper to demonstrate how filename templates correspond to run times. See also *EXAMPLE_INPUT_DIR*.

Used by: Example

EXTRACT_OUT_DIR

> **Warning:** **DEPRECATED:** Please use *EXTRACT_TILES_OUTPUT_DIR*.

EXTRACT_TILES_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: ExtractTiles

EXTRACT_TILES_DLAT
The latitude value, in degrees. Set to the value that defines the resolution of the data (in decimal degrees).

Used by: ExtractTiles

EXTRACT_TILES_DLON
The longitude value, in degrees. Set to the value that defines the resolution of the data (in decimal degrees).

Used by: ExtractTiles

EXTRACT_TILES_FILTER_OPTS

> **Warning:** **DEPRECATED:** Please use *TC_STAT_JOB_ARGS* instead. Control what options are passed to the METplus extract_tiles utility.

Used by: ExtractTiles

EXTRACT_TILES_FILTERED_OUTPUT_TEMPLATE

Warning: **DEPRECATED:** Please use *EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE* instead.

EXTRACT_TILES_GRID_INPUT_DIR

Warning: **DEPRECATED:** Please use *FCST_EXTRACT_TILES_INPUT_DIR* and *OBS_EXTRACT_TILES_INPUT_DIR* instead.

EXTRACT_TILES_LAT_ADJ
Specify a latitude adjustment, in degrees to be used in the analysis. In the ExtractTiles wrapper, this corresponds to the 2m portion of the 2n x 2m subregion tile.

Used by: ExtractTiles

EXTRACT_TILES_LON_ADJ
Specify a longitude adjustment, in degrees to be used in the analysis. In the ExtractTiles wrapper, this corresponds to the 2n portion of the 2n x 2m subregion tile.

Used by: ExtractTiles

EXTRACT_TILES_MTD_INPUT_DIR
Directory containing MTD output to be read by ExtractTiles.

Used by: ExtractTiles

EXTRACT_TILES_MTD_INPUT_TEMPLATE
Template used to specify a file generated by Mode Time Domain (MTD) to filter input data to be used in ExtractTiles. Must set either this variable OR *EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE* but not both.

Used by: ExtractTiles

EXTRACT_TILES_NLAT
The number of latitude points, set to a whole number. This defines the number of latitude points to incorporate into the subregion (density).

Used by: ExtractTiles

EXTRACT_TILES_NLON
The number of longitude points, set to a whole number. This defines the number of longitude points to incorporate into the subregion (density).

Used by: ExtractTiles

EXTRACT_TILES_OUTPUT_DIR
Set the output directory for the METplus extract_tiles utility.

Used by: ExtractTiles

EXTRACT_TILES_OVERWRITE_TRACK

> **Warning:** **DEPRECATED:** Please use *EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS* instead.

EXTRACT_TILES_PAIRS_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *EXTRACT_TILES_TC_STAT_INPUT_DIR* instead.

EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS
Specify whether to overwrite the track data or not. Acceptable values: yes/no

Used by: ExtractTiles

EXTRACT_TILES_STAT_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *EXTRACT_TILES_TC_STAT_INPUT_DIR* instead.

EXTRACT_TILES_STAT_INPUT_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE* instead.

EXTRACT_TILES_TC_STAT_INPUT_DIR
Directory containing TCStat output to be read by ExtractTiles.

Used by: ExtractTiles

EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE
Template used to specify the dump row output tcst file generated by TCStat to filter input data to be used in ExtractTiles. Example: {init?fmt=%Y%m%d_%H}/filter_{init?fmt=%Y%m%d_%H}.tcst Must set either this variable OR *EXTRACT_TILES_MTD_INPUT_TEMPLATE* but not both.

Used by: ExtractTiles

EXTRACT_TILES_VAR_LIST
Control what variables the METplus extract_tiles utility runs on. Additional filtering by summary (via the MET tc_stat tool). Please refer to the MET User's Guide (TC-STAT Tools) for all the available options for filtering by summary method in tc-stat. If no additional filtering is required, simply leave the value to *EXTRACT_TILES_FILTER_OPTS* blank/empty in the METplus configuration file.

Used by: ExtractTiles

FCST_<n>_FIELD_NAME

Warning: **DEPRECATED:** Please use *FCST_PCP_COMBINE_<n>_FIELD_NAME* where N >=1 instead.

FCST_ASCII_REGEX_LEAD

Warning: **DEPRECATED:** Please use *FCST_EXTRACT_TILES_PREFIX* instead.

FCST_DATA_INTERVAL

Warning: **DEPRECATED:**

FCST_ENSEMBLE_STAT_FILE_WINDOW_BEGIN
See *OBS_ENSEMBLE_STAT_FILE_WINDOW_BEGIN*

Used by:

FCST_ENSEMBLE_STAT_FILE_WINDOW_END
See *OBS_ENSEMBLE_STAT_FILE_WINDOW_END*

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_INPUT_DATATYPE
Specify the data type of the input directory for forecast files used with the MET ensemble_stat tool. Currently valid options are NETCDF, GRIB, and GEMPAK. If set to GEMPAK, data will automatically be converted to NetCDF via GempakToCF. Similar variables exists for observation grid and point data called *OBS_ENSEMBLE_STAT_INPUT_GRID_DATATYPE* and *OBS_ENSEMBLE_STAT_INPUT_POINT_DATATYPE*.

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_INPUT_DIR
Input directory for forecast files to use with the MET tool ensemble_stat. Corresponding variable exist for point and grid observation data called *OBS_ENSEMBLE_STAT_GRID_INPUT_DIR* and *OBS_ENSEMBLE_STAT_POINT_INPUT_DIR*.

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_INPUT_FILE_LIST
Specifies an explicit path to a file list file to pass ensembles into ensemble_stat. If set, *FCST_ENSEMBLE_STAT_INPUT_TEMPLATE* and *FCST_ENSEMBLE_STAT_INPUT_DIR* are ignored.

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_INPUT_TEMPLATE
Template used to specify forecast input filenames for the MET tool ensemble_stat. Corresponding variables exist for point and grid observation data called *OBS_ENSEMBLE_STAT_GRID_INPUT_TEMPLATE* and *OBS_ENSEMBLE_STAT_POINT_INPUT_TEMPLATE*. To utilize Python Embedding as input to the MET tools, set this value to PYTHON_NUMPY or PYTHON_XARRAY.

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_IS_PROB
Wrapper-specific version of *FCST_IS_PROB*.

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_PROB_IN_GRIB_PDS
Wrapper-specific version of *FCST_PROB_IN_GRIB_PDS*.

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_VAR<n>_LEVELS
Wrapper specific field info variable. See *FCST_VAR<n>_LEVELS*.

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_VAR<n>_NAME
Wrapper specific field info variable. See *FCST_VAR<n>_NAME*.

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_VAR<n>_OPTIONS
Wrapper specific field info variable. See *FCST_VAR<n>_OPTIONS*.

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_VAR<n>_THRESH
Wrapper specific field info variable. See *FCST_VAR<n>_THRESH*.

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_WINDOW_BEGIN
Passed to the EnsembleStat MET config file to determine the range of data within a file that should be used for processing forecast data. Units are seconds. If the variable is not set, EnsembleStat will use *FCST_WINDOW_BEGIN*.

Used by: EnsembleStat

FCST_ENSEMBLE_STAT_WINDOW_END
Passed to the EnsembleStat MET config file to determine the range of data within a file that should be used for processing forecast data. Units are seconds. If the variable is not set, ensemble_stat will use *FCST_WINDOW_END*.

Used by: EnsembleStat

FCST_EXACT_VALID_TIME

> **Warning:** **DEPRECATED:** No longer used. Please use *FCST_WINDOW_BEGIN* and *FCST_WINDOW_END* instead. If both of those variables are set to 0, the functionality is the same as FCST_EXACT_VALID_TIME = True.

FCST_EXTRACT_TILES_INPUT_DIR
Directory containing gridded forecast data to be used in ExtractTiles

Used by: ExtractTiles

FCST_EXTRACT_TILES_INPUT_TEMPLATE
Filename template used to identify forecast input file to ExtractTiles.

Used by: ExtractTiles

FCST_EXTRACT_TILES_OUTPUT_TEMPLATE
Filename template used to identify the forecast output file generated by ExtractTiles.

Used by: ExtractTiles

FCST_EXTRACT_TILES_PREFIX
Prefix for forecast tile files. Used to create filename of intermediate files that are created while performing a series analysis.

Used by: ExtractTiles

FCST_FILE_WINDOW_BEGIN
See *OBS_FILE_WINDOW_BEGIN*

Used by: EnsembleStat, GridStat, MODE, MTD, PB2NC, PointStat

FCST_FILE_WINDOW_END
See *OBS_FILE_WINDOW_END*

Used by: EnsembleStat, GridStat, MODE, MTD, PB2NC, PointStat

FCST_GEMPAK_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *GEMPAKTOCF_INPUT_DIR* instead.

FCST_GEMPAK_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *GEMPAKTOCF_INPUT_TEMPLATE* if GempakToCF is in the PROCESS_LIST.

FCST_GRID_STAT_FILE_TYPE
Specify the value for 'fcst.file_type' in the MET configuration file for GridStat.

Used by: GridStat

FCST_GRID_STAT_FILE_WINDOW_BEGIN
See *OBS_GRID_STAT_FILE_WINDOW_BEGIN*

Used by: GridStat

FCST_GRID_STAT_FILE_WINDOW_END
See *OBS_GRID_STAT_FILE_WINDOW_END*

Used by: GridStat

FCST_GRID_STAT_INPUT_DATATYPE
Specify the data type of the input directory for forecast files used with the MET grid_stat tool. Currently valid options are NETCDF, GRIB, and GEMPAK. If set to GEMPAK, data will automatically be converted to NetCDF via GempakToCF. A corresponding variable exists for observation data called *OBS_GRID_STAT_INPUT_DATATYPE*.

Used by: GridStat

FCST_GRID_STAT_INPUT_DIR
Input directory for forecast files to use with the MET tool grid_stat. A corresponding variable exists for

observation data called *OBS_GRID_STAT_INPUT_DIR*.

Used by: GridStat

FCST_GRID_STAT_INPUT_TEMPLATE
Template used to specify forecast input filenames for the MET tool grid_stat. A corresponding variable exists for observation data called *OBS_GRID_STAT_INPUT_TEMPLATE*. To utilize Python Embedding as input to the MET tools, set this value to PYTHON_NUMPY or PYTHON_XARRAY.

Used by: GridStat

FCST_GRID_STAT_IS_PROB
Wrapper-specific version of *FCST_IS_PROB*.

Used by: GridStat

FCST_GRID_STAT_PROB_IN_GRIB_PDS
Wrapper-specific version of *FCST_PROB_IN_GRIB_PDS*.

Used by: GridStat

FCST_GRID_STAT_PROB_THRESH
Threshold values to be used for probabilistic data in grid_stat. The value can be a single item or a comma separated list of items that must start with a comparison operator (>,>=,==,!=,<,<=,gt,ge,eq,ne,lt,le). A corresponding variable exists for observation data called *OBS_GRID_STAT_PROB_THRESH*.

Used by: GridStat

FCST_GRID_STAT_VAR<n>_LEVELS
Wrapper specific field info variable. See *FCST_VAR<n>_LEVELS*.

Used by: GridStat

FCST_GRID_STAT_VAR<n>_NAME
Wrapper specific field info variable. See *FCST_VAR<n>_NAME*.

Used by: GridStat

FCST_GRID_STAT_VAR<n>_OPTIONS
Wrapper specific field info variable. See *FCST_VAR<n>_OPTIONS*.

Used by: GridStat

FCST_GRID_STAT_VAR<n>_THRESH
Wrapper specific field info variable. See *FCST_VAR<n>_THRESH*.

Used by: GridStat

FCST_GRID_STAT_WINDOW_BEGIN
Passed to the GridStat MET config file to determine the range of data within a file that should be used for processing. Units are seconds. If the variable is not set, GridStat will use *FCST_WINDOW_BEGIN*.

Used by: GridStat

FCST_GRID_STAT_WINDOW_END
Passed to the GridStat MET config file to determine the range of data within a file that should be used for processing. Units are seconds. If the variable is not set, GridStat will use *FCST_WINDOW_END*.

Used by: GridStat

FCST_HR_END

> **Warning:** **DEPRECATED:** Please use *LEAD_SEQ* instead.

FCST_HR_INTERVAL

> **Warning:** **DEPRECATED:** Please use *LEAD_SEQ* instead.

FCST_HR_START

> **Warning:** **DEPRECATED:** Please use *LEAD_SEQ* instead.

FCST_INIT_HOUR_LIST
Specify a list of hours for initialization times of forecast files for use in the analysis.

Groups of values can be looped over by setting FCST_INIT_HOUR_LIST<n> and adding FCST_INIT_HOUR_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

FCST_INIT_INTERVAL

> **Warning: DEPRECATED:** Specify the stride for forecast initializations.

FCST_INPUT_DIR

> **Warning: DEPRECATED:** Please use FCST_[MET-APP]_INPUT_DIR` instead, i.e. *FCST_GRID_STAT_INPUT_DIR*

FCST_INPUT_DIR_REGEX

> **Warning: DEPRECATED:** Please use *FCST_POINT_STAT_INPUT_DIR* instead.

FCST_INPUT_FILE_REGEX

> **Warning: DEPRECATED:** Regular expression to use when identifying which forecast file to use.

FCST_INPUT_FILE_TMPL

> **Warning: DEPRECATED:** Please use *FCST_POINT_STAT_INPUT_TEMPLATE* instead.

FCST_IS_DAILY_FILE

> **Warning: DEPRECATED:**

FCST_IS_PROB
Boolean to specify whether the forecast data are probabilistic or not.

Used by: EnsembleStat, GridStat, MODE, MTD, PointStat, SeriesAnalysis

FCST_LEAD

> **Warning:** **DEPRECATED:** Please use *FCST_LEAD_LIST* instead.

FCST_LEAD_LIST
Specify the values of the FSCT_LEAD column in the MET .stat file to use. Comma separated list format, e.g.: 00, 24, 48, 72, 96, 120

Groups of values can be looped over by setting FCST_LEAD_LIST<n> and adding FCST_LEAD_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

FCST_LEVEL

> **Warning:** **DEPRECATED:** Please use *FCST_PCP_COMBINE_INPUT_ACCUMS* instead.

FCST_LEVEL_LIST
Specify the values of the FCST_LEV column in the MET .stat file to use.

Groups of values can be looped over by setting FCST_LEVEL_LIST<n> and adding FCST_LEVEL_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

FCST_MAX_FORECAST

> **Warning:** **DEPRECATED:** Please use *LEAD_SEQ_MAX* instead.

FCST_MIN_FORECAST

> **Warning:** **DEPRECATED:** Please use *FCST_PCP_COMBINE_MIN_FORECAST*.

FCST_MODE_CONV_RADIUS
Comma separated list of convolution radius values used by mode for forecast fields. A corresponding variable exists for observation data called *OBS_MODE_CONV_RADIUS*.

Used by: MODE

FCST_MODE_CONV_THRESH
Comma separated list of convolution threshold values used by mode for forecast fields. A corresponding variable exists for observation data called *OBS_MODE_CONV_THRESH*.

Used by: MODE

FCST_MODE_FILE_WINDOW_BEGIN
See *OBS_MODE_FILE_WINDOW_BEGIN*

Used by: MODE

FCST_MODE_FILE_WINDOW_END
See *OBS_MODE_FILE_WINDOW_END*

Used by: MODE

FCST_MODE_INPUT_DATATYPE
Specify the data type of the input directory for forecast files used with the MET mode tool. Currently valid options are NETCDF, GRIB, and GEMPAK. If set to GEMPAK, data will automatically be converted to NetCDF via GempakToCF. A corresponding variable exists for observation data called *OBS_MODE_INPUT_DATATYPE*.

Used by: MODE

FCST_MODE_INPUT_DIR
Input directory for forecast files to use with the MET tool mode. A corresponding variable exists for observation data called *OBS_MODE_INPUT_DIR*.

Used by: MODE

FCST_MODE_INPUT_TEMPLATE
Template used to specify forecast input filenames for the MET tool mode. A corresponding variable exists for observation data called *OBS_MODE_INPUT_TEMPLATE*. To utilize Python Embedding as input to the MET tools, set this value to PYTHON_NUMPY or PYTHON_XARRAY.

Used by: MODE

FCST_MODE_IS_PROB
Wrapper-specific version of *FCST_IS_PROB*.

Used by: MODE

FCST_MODE_MERGE_FLAG
Sets the merge_flag value in the mode config file for forecast fields. Valid values are NONE, THRESH, ENGINE, and BOTH. A corresponding variable exists for observation data called *OBS_MODE_MERGE_FLAG*.

Used by: MODE

FCST_MODE_MERGE_THRESH
Comma separated list of merge threshold values used by mode for forecast fields. A corresponding variable exists for observation data called *OBS_MODE_MERGE_THRESH*.

Used by: MODE

FCST_MODE_PROB_IN_GRIB_PDS
Wrapper-specific version of *FCST_PROB_IN_GRIB_PDS*.

Used by: MODE

FCST_MODE_VAR<n>_LEVELS
Wrapper specific field info variable. See *FCST_VAR<n>_LEVELS*.

Used by: MODE

FCST_MODE_VAR<n>_NAME
Wrapper specific field info variable. See *FCST_VAR<n>_NAME*.

Used by: MODE

FCST_MODE_VAR<n>_OPTIONS
Wrapper specific field info variable. See *FCST_VAR<n>_OPTIONS*.

Used by: MODE

FCST_MODE_VAR<n>_THRESH
Wrapper specific field info variable. See *FCST_VAR<n>_THRESH*.

Used by: MODE

FCST_MODE_WINDOW_BEGIN
Passed to the MODE MET config file to determine the range of data within a file that should be used for processing. Units are seconds. If the variable is not set, MODE will use *FCST_WINDOW_BEGIN*.

Used by: MODE

FCST_MODE_WINDOW_END
Passed to the MODE MET config file to determine the range of data within a file that should be used for processing. Units are seconds. If the variable is not set, MODE will use *FCST_WINDOW_END*.

Used by: MODE

FCST_MTD_CONV_RADIUS
Comma separated list of convolution radius values used by mode-TD for forecast files. A corresponding variable exists for observation data called *OBS_MTD_CONV_RADIUS*.

Used by:

FCST_MTD_CONV_THRESH
Comma separated list of convolution threshold values used by mode-TD for forecast files. A corresponding variable exists for observation data called *OBS_MTD_CONV_THRESH*.

Used by:

FCST_MTD_FILE_WINDOW_BEGIN
See *OBS_MTD_FILE_WINDOW_BEGIN*

Used by: MTD

FCST_MTD_FILE_WINDOW_END
See *OBS_MTD_FILE_WINDOW_END*

Used by: MTD

FCST_MTD_INPUT_DATATYPE
Specify the data type of the input directory for forecast files used with the MET mode-TD tool. Currently valid options are NETCDF, GRIB, and GEMPAK. If set to GEMPAK, data will automatically be converted to NetCDF via GempakToCF. A corresponding variable exists for observation data called *OBS_MTD_INPUT_DATATYPE*.

Used by: MTD

FCST_MTD_INPUT_DIR
Input directory for forecast files to use with the MET tool mode-TD. A corresponding variable exists for observation data called *OBS_MTD_INPUT_DIR*.

Used by: MTD

FCST_MTD_INPUT_FILE_LIST
Specifies an explicit path to a file list file to pass into mtd with the -fcst or -single argument. If set, *FCST_MTD_INPUT_TEMPLATE* and *FCST_MTD_INPUT_DIR* are ignored. See also *OBS_MTD_INPUT_FILE_LIST*.

Used by: MTD

FCST_MTD_INPUT_TEMPLATE
Template used to specify forecast input filenames for the MET tool mode-TD. A corresponding variable exists for observation data called *OBS_MTD_INPUT_TEMPLATE*. To utilize Python Embedding as input to the MET tools, set this value to PYTHON_NUMPY or PYTHON_XARRAY.

Used by: MTD

FCST_MTD_IS_PROB
Wrapper-specific version of *FCST_IS_PROB*.

Used by: MTD

FCST_MTD_PROB_IN_GRIB_PDS
Wrapper-specific version of *FCST_PROB_IN_GRIB_PDS*.

Used by: MTD

FCST_MTD_VAR<n>_LEVELS
Wrapper specific field info variable. See *FCST_VAR<n>_LEVELS*.

Used by: MTD

FCST_MTD_VAR<n>_NAME
Wrapper specific field info variable. See *FCST_VAR<n>_NAME*.

Used by: MTD

FCST_MTD_VAR<n>_OPTIONS
Wrapper specific field info variable. See *FCST_VAR<n>_OPTIONS*.

Used by: MTD

FCST_MTD_VAR<n>_THRESH
Wrapper specific field info variable. See *FCST_VAR<n>_THRESH*.

Used by: MTD

FCST_NATIVE_DATA_TYPE

> **Warning:** **DEPRECATED:** Please use *FCST_PCP_COMBINE_INPUT_DATATYPE* instead

FCST_NC_TILE_REGEX

> **Warning:** **DEPRECATED:** Please use *FCST_EXTRACT_TILES_PREFIX* instead.

FCST_PCP_COMBINE_<n>_FIELD_NAME

> **Warning:** **DEPRECATED:** Please use *FCST_PCP_COMBINE_INPUT_NAMES* instead.

FCST_PCP_COMBINE_BUCKET_INTERVAL
Used when *FCST_PCP_COMBINE_INPUT_ACCUMS* contains {lead} in the list. This is the interval to reset the bucket accumulation. For example, if the accumulation is reset every 3 hours (forecast 1

hour has 1 hour accum, forecast 2 hour has 2 hour accum, forecast 3 hour has 3 hour accum, forecast 4 hour has 1 hour accum, etc.) then this should be set to 3 or 3H. Units are assumed to be hours unless specified with Y, m, d, H, M, or S.

Used by: PCPCombine

FCST_PCP_COMBINE_COMMAND

Used only when *FCST_PCP_COMBINE_METHOD* = USER_DEFINED. Custom command to run PCP-Combine with a complex call that doesn't fit common use cases. Value can include filename template syntax, i.e. {valid?fmt=%Y%m%d}, that will be substituted based on the current runtime. The name of the application and verbosity flag does not need to be included. For example, if set to '-derive min,max /some/file' the command run will be pcp_combine -v 2 -derive min,max /some/file. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_COMMAND*.

Used by: PCPCombine

FCST_PCP_COMBINE_CONSTANT_INIT

If True, only look for forecast files that have a given initialization time. Used only if *FCST_PCP_COMBINE_INPUT_TEMPLATE* has a 'lead' tag. If set to False, the lowest forecast lead for each search (valid) time is used. See *OBS_PCP_COMBINE_CONSTANT_INIT*

Used by: PCPCombine

FCST_PCP_COMBINE_DATA_INTERVAL

> **Warning: DEPRECATED:**

FCST_PCP_COMBINE_DERIVE_LOOKBACK

> **Warning: DEPRECATED:** Please use *FCST_PCP_COMBINE_LOOKBACK* instead.

FCST_PCP_COMBINE_EXTRA_LEVELS

Specify a list of any additional fields to add to the command. The items in this list correspond to the list set by *FCST_PCP_COMBINE_EXTRA_NAMES*. If this list has fewer items than the names list, then no level value will be specified for those names (i.e. if using Python Embedding). A corresponding variable exists for observation data called *OBS_PCP_COMBINE_EXTRA_LEVELS*. See *FCST_PCP_COMBINE_EXTRA_NAMES* for an example.

Used by: PCPCombine

FCST_PCP_COMBINE_EXTRA_NAMES

Specify a list of any additional fields to add to the command. The items in this list correspond to the list set by *FCST_PCP_COMBINE_EXTRA_LEVELS*. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_EXTRA_NAMES*. Example:

FCST_PCP_COMBINE_EXTRA_NAMES = TMP, HGT
FCST_PCP_COMBINE_EXTRA_LEVELS = "(,)", "(,)"

This will add the following to the end of the command:

-field 'name="TMP"; level="(,)";' -field 'name="HGT"; level="(,)";'

Used by: PCPCombine

FCST_PCP_COMBINE_EXTRA_OUTPUT_NAMES

Specify a list of output names for any additional fields to add to the command. The items in this list correspond to the list set by *FCST_PCP_COMBINE_EXTRA_NAMES*. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_EXTRA_OUTPUT_NAMES*. Example:

Used by: PCPCombine

FCST_PCP_COMBINE_INPUT_ACCUMS

Specify what accumulation levels should be used from the forecast data for the analysis. This is a list of input accumulations in the order of preference to use to build the desired accumulation. If an accumulation cannot be used (i.e. it is larger than the remaining accumulation that needs to be built) then the next value in the list is tried. Units are assumed to be hours unless a time identifier such as Y, m, d, H, M, S is specifed at the end of the value, i.e. 30M or 1m.

If the name and/or level of the accumulation value must be specified for the data, then a list of equal length to this variable must be set for *FCST_PCP_COMBINE_INPUT_NAMES* and *FCST_PCP_COMBINE_INPUT_LEVELS*. See this sections for more information.

This variable can be set to {lead} if the accumulation found in a given file corresponds to the forecast lead of the data. If this is the case, *FCST_PCP_COMBINE_BUCKET_INTERVAL* can be used to reset the accumulation at a given interval.

A corresponding variable exists for observation data called *OBS_PCP_COMBINE_INPUT_ACCUMS*.

Examples:

1H, 30M

This will attempt to use a 1 hour accumulation, then try to use a 30 minute accumulation if the first value did not succeed.

Used by: PCPCombine

FCST_PCP_COMBINE_INPUT_DATATYPE

Specify the data type of the input directory for forecast files used with the MET pcp_combine tool. Currently valid options are NETCDF, GRIB, and GEMPAK. Required by pcp_combine if *FCST_PCP_COMBINE_RUN* is True. Replaces deprecated variable *FCST_NATIVE_DATA_TYPE*. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_INPUT_DATATYPE*.

Used by: PCPCombine

FCST_PCP_COMBINE_INPUT_DIR

Specify the input directory for forecast files used with the MET pcp_combine tool. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_INPUT_DIR*.

Used by: PCPCombine

FCST_PCP_COMBINE_INPUT_LEVEL

Warning: DEPRECATED: Please use *FCST_PCP_COMBINE_INPUT_ACCUMS*.

FCST_PCP_COMBINE_INPUT_LEVELS

Specify which levels correspond to each accumulation specifed in FCST_PCP_COMBINE_INPUT_ACCUMS for the forecast data for the analysis. See *FCST_PCP_COMBINE_INPUT_ACCUMS* for more information. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_INPUT_LEVELS*. Examples:

FCST_PCP_COMBINE_INPUT_ACCUMS = 1
FCST_PCP_COMBINE_INPUT_NAMES = P01M_NONE
FCST_PCP_COMBINE_INPUT_LEVELS = "(0,*,*)"

This says that the 1 hour accumulation field name is P01M_NONE and the level (0,*,*), which is NetCDF format to specify the first item of the first dimension.

Used by: PCPCombine

FCST_PCP_COMBINE_INPUT_NAMES

Specify which field names correspond to each accumulation specifed in FCST_PCP_COMBINE_INPUT_ACCUMS for the forecast data for the analysis. See *FCST_PCP_COMBINE_INPUT_ACCUMS* for more information. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_INPUT_NAMES*. Examples:

FCST_PCP_COMBINE_INPUT_ACCUMS = 6, 1
FCST_PCP_COMBINE_INPUT_NAMES = P06M_NONE, P01M_NONE

This says that the 6 hour accumulation field name is P06M_NONE and the 1 hour accumulation field name is P01M_NONE.

To utilize Python Embedding as input to the MET tools, set this value to the python script command with arguments. This value can include filename template syntax such as {valid?fmt=%Y%m%d%H}.

Used by: PCPCombine

FCST_PCP_COMBINE_INPUT_OPTIONS

Specify optional additional options that correspond to each accumulation specifed in FCST_PCP_COMBINE_INPUT_ACCUMS for the forecast data for the analysis. See *FCST_PCP_COMBINE_INPUT_ACCUMS* for more information. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_INPUT_OPTIONS*. Examples:

FCST_PCP_COMBINE_INPUT_ACCUMS = 6, 1
FCST_PCP_COMBINE_INPUT_NAMES = P06M_NONE, P01M_NONE
FCST_PCP_COMBINE_INPUT_OPTIONS = something = else;, another_thing = else;

Used by: PCPCombine

FCST_PCP_COMBINE_INPUT_TEMPLATE

Template used to specify input filenames for forecast files used by the MET pcp_combine tool. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_INPUT_TEMPLATE*. To utilize Python Embedding as input to the MET tools, set this value to PYTHON_NUMPY or PYTHON_XARRAY.

Used by: PCPCombine

FCST_PCP_COMBINE_IS_DAILY_FILE

Warning: DEPRECATED:

FCST_PCP_COMBINE_LOOKBACK
Specify how far to look back in time to find files for building commands to run the pcp_combine tool. If processing precipitation accumulation data, this is equivalent to the desired output accumulation to compute. Units are assumed to be hours unless a time identifier such as Y, m, d, H, M, S is specified at the end of the value, i.e. 30M or 1m. If unset, *FCST_PCP_COMBINE_OUTPUT_ACCUM* will be used. If that is unset, then *FCST_PCP_COMBINE_DERIVE_LOOKBACK* will be used. If none of the variables are set or set to 0, data will be obtained by using the input template with the current runtime instead of looking backwards in time. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_LOOKBACK*.

Used by: PCPCombine

FCST_PCP_COMBINE_MAX_FORECAST
Specify the maximum forecast lead time to use when finding the lowest forecast lead to use in pcp_combine. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_MAX_FORECAST*.

Used by: PCPCombine

FCST_PCP_COMBINE_METHOD
Specify the method to be used with the MET pcp_combine tool processing forecast data.Valid options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_METHOD*.

Used by: PCPCombine

FCST_PCP_COMBINE_MIN_FORECAST
Specify the minimum forecast lead time to use when finding the lowest forecast lead to use in pcp_combine. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_MIN_FORECAST*.

Used by: PCPCombine

FCST_PCP_COMBINE_OUTPUT_ACCUM
Specify desired accumulation to be built from the forecast data. Synonym for *FCST_PCP_COMBINE_LOOKBACK*.

A corresponding variable exists for observation data called *OBS_PCP_COMBINE_OUTPUT_ACCUM*.

Used by: PCPCombine

FCST_PCP_COMBINE_OUTPUT_DIR
Specify the output directory for forecast files generated by the MET pcp_combine tool. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_OUTPUT_DIR*.

Used by: PCPCombine

FCST_PCP_COMBINE_OUTPUT_NAME
Specify the output field name from processing forecast data. If this variable is not set, then *FCST_VAR<n>_NAME* is used.

A corresponding variable exists for observation data called *OBS_PCP_COMBINE_OUTPUT_NAME*.

Example: APCP

Used by: PCPCombine

FCST_PCP_COMBINE_OUTPUT_TEMPLATE
Template used to specify output filenames for forecast files generated by the MET pcp_combine tool. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_OUTPUT_TEMPLATE*. To utilize Python Embedding as input to the MET tools, set this value to PYTHON_NUMPY or PYTHON_XARRAY.

Used by: PCPCombine

FCST_PCP_COMBINE_RUN
Specify whether to run the MET pcp_combine tool on forecast data or not. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_RUN*.Acceptable values: true/false

Used by: PCPCombine

FCST_PCP_COMBINE_STAT_LIST
List of statistics to process when using the MET pcp_combine tool on forecast data in derive mode. A corresponding variable exists for observation data called *OBS_PCP_COMBINE_STAT_LIST*.Acceptable values: sum, min, max, range, mean, stdev, vld_count

Used by: PCPCombine

FCST_PCP_COMBINE_TIMES_PER_FILE

> **Warning: DEPRECATED:**

FCST_PCP_COMBINE_USE_ZERO_ACCUM
Only used if running PCPCombine wrapper with *FCST_PCP_COMBINE_METHOD* = SUBTRACT. If True, build a -subtract command using the 0 accumulation as the 2nd input. If False (default), instead build an -add command with a single input if the 2nd input is a 0 accumulation.

Used by: PCPCombine

FCST_POINT_STAT_FILE_WINDOW_BEGIN
See *OBS_POINT_STAT_FILE_WINDOW_BEGIN*

Used by: PointStat

FCST_POINT_STAT_FILE_WINDOW_END
See *OBS_POINT_STAT_FILE_WINDOW_END*

Used by: PointStat

FCST_POINT_STAT_INPUT_DATATYPE
Specify the data type of the input directory for forecast files used with the MET point_stat tool. Currently valid options are NETCDF, GRIB, and GEMPAK. If set to GEMPAK, data will automatically be converted to NetCDF via GempakToCF. A corresponding variable exists for observation data called *OBS_POINT_STAT_INPUT_DATATYPE*.

Used by: PointStat

FCST_POINT_STAT_INPUT_DIR
Input directory for forecast files to use with the MET tool point_stat. A corresponding variable exists for observation data called *OBS_POINT_STAT_INPUT_DIR*.

Used by: PointStat

FCST_POINT_STAT_INPUT_TEMPLATE
Template used to specify forecast input filenames for the MET tool point_stat. A corresponding variable exists for observation data called *OBS_POINT_STAT_INPUT_TEMPLATE*. To utilize Python Embedding as input to the MET tools, set this value to PYTHON_NUMPY or PYTHON_XARRAY.

Used by: GriPointStat

FCST_POINT_STAT_IS_PROB
Wrapper-specific version of *FCST_IS_PROB*.

Used by: PointStat

FCST_POINT_STAT_PROB_IN_GRIB_PDS
Wrapper-specific version of *FCST_PROB_IN_GRIB_PDS*.

Used by: PointStat

FCST_POINT_STAT_VAR<n>_LEVELS
Wrapper specific field info variable. See *FCST_VAR<n>_LEVELS*.

Used by: PointStat

FCST_POINT_STAT_VAR<n>_NAME
Wrapper specific field info variable. See *FCST_VAR<n>_NAME*.

Used by: PointStat

FCST_POINT_STAT_VAR<n>_OPTIONS
Wrapper specific field info variable. See *FCST_VAR<n>_OPTIONS*.

Used by: PointStat

FCST_POINT_STAT_VAR<n>_THRESH
Wrapper specific field info variable. See *FCST_VAR<n>_THRESH*.

Used by: PointStat

FCST_POINT_STAT_WINDOW_BEGIN
Passed to the PointStat MET config file to determine the range of data within a file that should be

used for processing forecast data. Units are seconds. If the variable is not set, PointStat will use *OBS_WINDOW_BEGIN*.

Used by: PointStat

FCST_POINT_STAT_WINDOW_END
Passed to the PointStat MET config file to determine the range of data within a file that should be used for processing forecast data. Units are seconds. If the variable is not set, PointStat will use *OBS_WINDOW_END*.

Used by: PointStat

FCST_PROB_IN_GRIB_PDS
Boolean to specify whether the probabilistic forecast data is stored in the GRIB Product Definition Section or not. Only used when *FCST_IS_PROB* is True.

Used by: EnsembleStat, GridStat, MODE, MTD, PointStat, SeriesAnalysis

FCST_REGRID_DATA_PLANE_INPUT_DATATYPE
Specify the data type of the input directory for forecast files used with the MET regrid_data_plane tool. Currently valid options are NETCDF, GRIB, and GEMPAK. Required by pcp_combine. A corresponding variable exists for observation data called *OBS_REGRID_DATA_PLANE_INPUT_DATATYPE*.

Used by: RegridDataPlane

FCST_REGRID_DATA_PLANE_INPUT_DIR
Specify the input directory for forecast files used with the MET regrid_data_plane tool. A corresponding variable exists for observation data called *OBS_REGRID_DATA_PLANE_INPUT_DIR*.

Used by: RegridDataPlane

FCST_REGRID_DATA_PLANE_INPUT_TEMPLATE
Template used to specify input filenames for forecast data used by the MET regrid_data_plane tool. It not set, METplus will use *FCST_REGRID_DATA_PLANE_TEMPLATE*. A corresponding variable exists for observation data called *OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE*. To utilize Python Embedding as input to the MET tools, set this value to PYTHON_NUMPY or PYTHON_XARRAY.

Used by: RegridDataPlane

FCST_REGRID_DATA_PLANE_OUTPUT_DIR
Specify the output directory for forecast files used with the MET regrid_data_plane tool. A corresponding variable exists for observation data called *OBS_REGRID_DATA_PLANE_OUTPUT_DIR*.

Used by: RegridDataPlane

FCST_REGRID_DATA_PLANE_OUTPUT_TEMPLATE
Template used to specify output filenames for forecast data used by the MET regrid_data_plane tool. It not set, METplus will use *FCST_REGRID_DATA_PLANE_TEMPLATE*. A corresponding variable exists for observation data called *OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE*.

Used by: RegridDataPlane

FCST_REGRID_DATA_PLANE_RUN
If True, process forecast data with RegridDataPlane.

Used by: RegridDataPlane

FCST_REGRID_DATA_PLANE_TEMPLATE
Template used to specify filenames for forecast data used by the MET regrid_data_plane tool. To specify different templates for input and output files , use *FCST_REGRID_DATA_PLANE_INPUT_TEMPLATE* and *FCST_REGRID_DATA_PLANE_OUTPUT_TEMPLATE*. A corresponding variable exists for observation data called *OBS_REGRID_DATA_PLANE_TEMPLATE*.

Used by: RegridDataPlane

FCST_REGRID_DATA_PLANE_VAR<n>_INPUT_FIELD_NAME
Specify the (optional) forecast input field name that is read by RegridDataPlane. The name corresponds to *FCST_VAR<n>_NAME*. This is used when using Python Embedding as input to the MET tool, because the *FCST_VAR<n>_NAME* defines the python script to call.

Used by: RegridDataPlane

FCST_REGRID_DATA_PLANE_VAR<n>_INPUT_LEVEL
Specify the (optional) forecast input field level that is read by RegridDataPlane. The name corresponds

to *FCST_VAR<n>_LEVELS*. This is used when using Python Embedding as input to the MET tool, because the *FCST_VAR<n>_LEVELS* defines the python script to call.

Used by: RegridDataPlane

FCST_REGRID_DATA_PLANE_VAR<n>_OUTPUT_FIELD_NAME
Specify the forecast output field name that is created by RegridDataPlane. The name corresponds to *FCST_VAR<n>_NAME*. This is used when using Python Embedding as input to the MET tool, because the *FCST_VAR<n>_NAME* defines the python script to call.

Used by: RegridDataPlane

FCST_SERIES_ANALYSIS_ASCII_REGEX_LEAD

Warning: **DEPRECATED:** Please use *FCST_EXTRACT_TILES_PREFIX* instead.

FCST_SERIES_ANALYSIS_CAT_THRESH
Specify the value for 'fcst.cat_thresh' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

FCST_SERIES_ANALYSIS_INPUT_DATATYPE
Set the file_type entry of the fcst dictionary in the MET config file for SeriesAnalysis.

Used by: SeriesAnalysis

FCST_SERIES_ANALYSIS_INPUT_DIR
Specify the directory to read forecast input in SeriesAnalysis. See also *FCST_SERIES_ANALYSIS_INPUT_TEMPLATE*

Used by: SeriesAnalysis

FCST_SERIES_ANALYSIS_INPUT_FILE_LIST
Specifies an explicit path to a file list file to pass into series_analysis with the -fcst argument. If set, *OBS_SERIES_ANALYSIS_INPUT_FILE_LIST* must also be set and

FCST_SERIES_ANALYSIS_INPUT_TEMPLATE and *FCST_SERIES_ANALYSIS_INPUT_DIR* are ignored. See also *BOTH_SERIES_ANALYSIS_INPUT_FILE_LIST*.

Used by: SeriesAnalysis

FCST_SERIES_ANALYSIS_INPUT_TEMPLATE
Template to find forecast input in SeriesAnalysis. See also *FCST_SERIES_ANALYSIS_INPUT_DIR*

Used by: SeriesAnalysis

FCST_SERIES_ANALYSIS_IS_PROB
Wrapper-specific version of *FCST_IS_PROB*.

Used by: SeriesAnalysis

FCST_SERIES_ANALYSIS_NC_TILE_REGEX

Warning: **DEPRECATED:** Please use *FCST_EXTRACT_TILES_PREFIX* instead.

FCST_SERIES_ANALYSIS_PROB_IN_GRIB_PDS
Wrapper-specific version of *FCST_PROB_IN_GRIB_PDS*.

Used by: SeriesAnalysis

FCST_SERIES_ANALYSIS_PROB_THRESH
Threshold values to be used for probabilistic data in series_analysis. The value can be a single item or a comma separated list of items that must start with a comparison operator (>,>=,==,!=,<,<=,gt,ge,eq,ne,lt,le).

Used by: SeriesAnalysis

FCST_SERIES_ANALYSIS_TILE_INPUT_DIR

Warning: **DEPRECATED:** Please use *FCST_SERIES_ANALYSIS_INPUT_DIR* instead.

FCST_THRESH

Warning: **DEPRECATED:** Please use *FCST_THRESH_LIST* instead.

FCST_THRESH_LIST
Specify the values of the FCST_THRESH column in the MET .stat file to use.

Groups of values can be looped over by setting FCST_THRESH_LIST<n> and adding FCST_THRESH_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

FCST_TILE_PREFIX

Warning: DEPRECATED: Please use *FCST_EXTRACT_TILES_PREFIX* instead.

FCST_TILE_REGEX

Warning: DEPRECATED: No longer used. Regular expression for forecast input files that are in GRIB2.

FCST_TIMES_PER_FILE

Warning: DEPRECATED:

FCST_UNITS_LIST
Specify the values of the FCST_UNITS column in the MET .stat file to use.

Groups of values can be looped over by setting FCST_UNITS_LIST<n> and adding FCST_UNITS_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

FCST_VALID_HOUR_LIST
Specify a list of hours for valid times of forecast files for use in the analysis.

Groups of values can be looped over by setting FCST_VALID_HOUR_LIST<n> and adding FCST_VALID_HOUR_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

FCST_VAR

> **Warning: DEPRECATED:** No longer used.

FCST_VAR<n>_LEVELS

Define the levels for the <n>th forecast variable to be used in the analysis where <n> is an integer >= 1. The value can be a single item or a comma separated list of items. You can define NetCDF levels, such as (0,*,*), but you will need to surround these values with quotation marks so that the commas in the item are not interpreted as an item delimeter. Some examples:

FCST_VAR1_LEVELS = A06, P500
FCST_VAR2_LEVELS ="(0,*,*),(1,*,*)"

There can be <n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.:

FCST_VAR1_LEVELS
FCST_VAR2_LEVELS
...
FCST_VAR<n>_LEVELS

If FCST_VAR<n>_LEVELS is set, then *OBS_VAR<n>_LEVELS* must be set as well. If the same value applies to both forecast and observation data, use *BOTH_VAR<n>_LEVELS*.

See *Field Info* (page 58) for more information.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

FCST_VAR<n>_NAME

Define the name for the <n>th forecast variable to be used in the analysis where <n> is an integer >= 1. If *FCST_VAR<n>_NAME* is set, then *OBS_VAR<n>_NAME* must be set. If the same value applies to both forecast and observation data, use *BOTH_VAR<n>_NAME*. There can be s<n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.:

FCST_VAR1_NAME
FCST_VAR2_NAME
...
FCST_VAR<n>_NAME

See *Field Info* (page 58) for more information.

This value can be set to a call to a python script with arguments to supply data to the MET tools via Python Embedding. Filename template syntax can be used here to specify time information of an input file, i.e. {valid?fmt=%Y%m%d%H}. See the MET User's Guide for more information about Python Embedding in the MET tools.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

FCST_VAR<n>_OPTIONS

Define the options for the <n>th forecast variable to be used in the analysis where <n> is an integer >= 1. These addition options will be applied to every name/level/threshold combination for VAR<n>. There can be <n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.:

FCST_VAR1_OPTIONS
FCST_VAR2_OPTIONS
...
FCST_VAR<n>_OPTIONS

See *Field Info* (page 58) for more information.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

FCST_VAR<n>_THRESH

Define the threshold(s) for the <n>th forecast variable to be used in the analysis where <n> is an integer >= 1. The value can be a single item or a comma separated list of items that must start with a comparison operator (>,>=,==,!=,<,<=,gt,ge,eq,ne,lt,le). If *FCST_VAR<n>_THRESH* is not set but *OBS_VAR<n>_THRESH* is, the same information will be used for both variables. There can be <n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.: | FCST_VAR1_THRESH | FCST_VAR2_THRESH | ... | FCST_VAR<n>_THRESH

If *FCST_VAR<n>_THRESH* is set, then *OBS_VAR<n>_THRESH* must be set as well. If the same value applies to both forecast and observation data, use *BOTH_VAR<n>_THRESH*.

See *Field Info* (page 58) for more information.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

FCST_VAR_LEVEL

Warning: **DEPRECATED:** Please use *FCST_LEVEL_LIST* instead.

FCST_VAR_LIST

Specify the values of the FCST_VAR column in the MET .stat file to use.

Groups of values can be looped over by setting FCST_VAR_LIST<n> and adding FCST_VAR_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

FCST_VAR_NAME

Warning: **DEPRECATED:** Please use *FCST_VAR_LIST* instead.

FCST_WINDOW_BEGIN

See *OBS_WINDOW_BEGIN*

Used by: EnsembleStat, GridStat, MODE, MTD, PB2NC, PointStat

FCST_WINDOW_END

See *OBS_WINDOW_END*

Used by: EnsembleStat, GridStat, MODE, MTD, PB2NC, PointStat

FHR_BEG

Warning: **DEPRECATED:** Please use *LEAD_SEQ* instead.

FHR_END

Warning: **DEPRECATED:** Please use *LEAD_SEQ* instead.

FHR_GROUP_BEG

Warning: **DEPRECATED:** Please use *LEAD_SEQ_<n>* instead.

FHR_GROUP_END

Warning: **DEPRECATED:** Please use *LEAD_SEQ_<n>* instead.

FHR_GROUP_LABELS

Warning: **DEPRECATED:** Please use *LEAD_SEQ_<n>_LABEL* instead.

FHR_INC

Warning: **DEPRECATED:** Please use *LEAD_SEQ* instead.

FILE_LISTS_DIR
Directory to store text files generated by METplus that contain a list of input file paths to pass in a MET executable that allows multiple input files. By default this directory is found under the *STAGING_DIR* and contains the *LOG_TIMESTAMP* to easily identify which file lists were generated from a METplus run.

Used by: All

FILE_WINDOW_BEGIN
Used to control the lower bound of the window around the valid time to determine if a file should be used for processing. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. This value will be used for all wrappers that look for all files unless it is overridden by a wrapper specific configuration variable. For example, if *OBS_GRID_STAT_FILE_WINDOW_BEGIN* is set, the GridStat wrapper will use that value. If *PB2NC_FILE_WINDOW_BEGIN* is not set, then the PB2NC wrapper will use *OBS_FILE_WINDOW_BEGIN*. If *OBS_FILE_WINDOW_BEGIN* is not set, it will use FILE_WINDOW_BEGIN if it is set. If not, it will default to 0. If the begin and end file window values are both 0, then only a file matching the exact run time will be considered.

Used by: All

FILE_WINDOW_END
Used to control the upper bound of the window around the valid time to determine if a file should

be used for processing. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. This value will be used for all wrappers that look for all files unless it is overridden by a wrapper specific configuration variable. For example, if *OBS_GRID_STAT_FILE_WINDOW_END* is set, the GridStat wrapper will use that value. If *PB2NC_FILE_WINDOW_END* is not set, then the PB2NC wrapper will use *OBS_FILE_WINDOW_END*. If *OBS_FILE_WINDOW_END* is not set, it will use FILE_WINDOW_END if it is set. If not, it will default to 0. If the begin and end file window values are both 0, then only a file matching the exact run time will be considered.

Used by: All

FILTER

Warning: DEPRECATED: Please use *TCMPR_PLOTTER_FILTER* instead.

FILTERED_TCST_DATA_FILE

Warning: DEPRECATED: Please use *TCMPR_PLOTTER_FILTERED_TCST_DATA_FILE* instead.

FOOTNOTE_FLAG

Warning: DEPRECATED: Please use *TCMPR_PLOTTER_FOOTNOTE_FLAG* instead.

FORECAST_TMPL

Warning: DEPRECATED: Please use *TC_PAIRS_ADECK_TEMPLATE*.

GEMPAKTOCF_CLASSPATH

Warning: DEPRECATED: Please use *GEMPAKTOCF_JAR* instead. Path to the GempakToCF binary file and the NetCDF jar file required to run GempakToCF.

GEMPAKTOCF_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: GempakToCF

GEMPAKTOCF_INPUT_DIR
Specify the input directory for the tool used to convert GEMPAK files to netCDF.

Used by: GempakToCF

GEMPAKTOCF_INPUT_TEMPLATE
Filename template used for input files to the tool used to convert GEMPAK files to netCDF.

Used by: GempakToCF

GEMPAKTOCF_JAR
Path to the GempakToCF.jar file to run GempakToCF. The tool is available on the MET webpage here: https://dtcenter.org/sites/default/files/community-code/metplus/utilities/GempakToCF.jar. Must be set if running GempakToCF wrapper, if using a filename template that ends with .grd, or if specifying an *_INPUT_DATATYPE item as GEMPAK.

Used by: GempakToCF, other wrappers that will read Gempak data

GEMPAKTOCF_OUTPUT_DIR
Specify the output directory for files generated by the tool used to convert GEMPAK files to netCDF.

Used by: GempakToCF

GEMPAKTOCF_OUTPUT_TEMPLATE
Filename template used for output files from the tool used to convert GEMPAK files to netCDF.

Used by: GempakToCF

GEMPAKTOCF_SKIP_IF_OUTPUT_EXISTS
If True, do not run GempakToCF if output file already exists. Set to False to overwrite files.

Used by: GempakToCF

GEN_ENS_PROD_CAT_THRESH
Specify the value for 'cat_thresh' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CENSOR_THRESH
Specify the value for 'censor_thresh' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CENSOR_VAL
Specify the value for 'censor_val' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_DAY_INTERVAL
Specify the value for 'climo_mean.day_interval' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_FIELD
See: *<TOOL-NAME>_CLIMO_MEAN_FIELD*

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_FILE_NAME
Specify the value for 'climo_mean.file_name' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_HOUR_INTERVAL
Specify the value for 'climo_mean.hour_interval' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_MATCH_MONTH
Specify the value for 'climo_mean.match_month' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_REGRID_METHOD
Specify the value for 'climo_mean.regrid.method' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_REGRID_SHAPE
Specify the value for 'climo_mean.regrid.shape' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_REGRID_VLD_THRESH
Specify the value for 'climo_mean.regrid.vld_thresh' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_REGRID_WIDTH
Specify the value for 'climo_mean.regrid.width' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_TIME_INTERP_METHOD
Specify the value for 'climo_mean.time_interp_method' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_USE_FCST
If set to True, use the field array from the fcst dictionary for the climo_mean fields for GenEnsProd. Sets "climo_mean = fcst;" in the wrapped MET config file. Only used if *GEN_ENS_PROD_CLIMO_MEAN_FIELD* is unset. See also *GEN_ENS_PROD_CLIMO_MEAN_USE_OBS*.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_USE_OBS
If set to True, use the field array from the obs dictionary for the climo_mean fields for GenEnsProd. Sets "climo_mean = obs;" in the wrapped MET config file. Only used if *GEN_ENS_PROD_CLIMO_MEAN_FIELD* is unset. See also *GEN_ENS_PROD_CLIMO_MEAN_USE_FCST*.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_VAR<n>_LEVELS
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_LEVELS*

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_VAR<n>_NAME
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_NAME*

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_MEAN_VAR<n>_OPTIONS
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_OPTIONS*

GEN_ENS_PROD_CLIMO_STDEV_DAY_INTERVAL
Specify the value for 'climo_stdev.day_interval' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_FIELD
Specify the value for 'climo_stdev.field' in the MET configuration file for GenEnsProd. The value set here must include the proper formatting that is expected in MET configuration file for specifying field information. Example: {name="TMP"; level="(,)";} To set the field information un-formatted, use the *GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_NAME*, *GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_LEVELS*, and *GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_OPTIONS* variables.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_FILE_NAME
Specify the value for 'climo_stdev.file_name' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_HOUR_INTERVAL
Specify the value for 'climo_stdev.hour_interval' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_MATCH_MONTH
Specify the value for 'climo_stdev.match_month' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_REGRID_METHOD
Specify the value for 'climo_stdev.regrid.method' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_REGRID_SHAPE
Specify the value for 'climo_stdev.regrid.shape' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_REGRID_VLD_THRESH
Specify the value for 'climo_stdev.regrid.vld_thresh' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_REGRID_WIDTH
Specify the value for 'climo_stdev.regrid.width' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_TIME_INTERP_METHOD
Specify the value for 'climo_stdev.time_interp_method' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_USE_FCST
If set to True, use the field array from the fcst dictionary for the climo_stdev fields for

GenEnsProd. Sets "climo_stdev = fcst;" in the wrapped MET config file. Only used if *GEN_ENS_PROD_CLIMO_STDEV_FIELD* is unset. See also *GEN_ENS_PROD_CLIMO_STDEV_USE_OBS*.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_USE_OBS

If set to True, use the field array from the obs dictionary for the climo_stdev fields for GenEnsProd. Sets "climo_stdev = obs;" in the wrapped MET config file. Only used if *GEN_ENS_PROD_CLIMO_STDEV_FIELD* is unset. See also *GEN_ENS_PROD_CLIMO_STDEV_USE_FCST*.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_LEVELS

Specify the level of the nth field for 'climo_stdev.field' in the MET configuration file for GenEnsProd. If any fields are set using this variable, then *GEN_ENS_PROD_CLIMO_STDEV_FIELD* will be ignored. See also *GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_NAME* and *GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_NAME

Specify the name of the nth field for 'climo_stdev.field' in the MET configuration file for GenEnsProd. If any fields are set using this variable, then *GEN_ENS_PROD_CLIMO_STDEV_FIELD* will be ignored. See also *GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_LEVELS* and *GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: GenEnsProd

GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_OPTIONS

Specify the extra options of the nth field for 'climo_stdev.field' in the MET configuration file for GenEnsProd. If any fields are set using this variable, then *GEN_ENS_PROD_CLIMO_STDEV_FIELD* will be ignored. See also *GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_NAME* and *GEN_ENS_PROD_CLIMO_STDEV_VAR<n>_LEVELS*.

GEN_ENS_PROD_CONTROL_ID

Specify the value for 'control_id' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_CTRL_INPUT_DIR
Input directory for optional control file to use with the MET tool gen_ens_prod.

Used by: GenEnsProd

GEN_ENS_PROD_CTRL_INPUT_TEMPLATE
Template used to specify an optional control filename for the MET tool gen_ens_prod.

Used by: GenEnsProd

GEN_ENS_PROD_DESC
Specify the value for 'desc' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENS_FILE_TYPE
Specify the value for 'ens.file_type' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENS_MEMBER_IDS
Specify the value for 'ens_member_ids' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENS_THRESH
Specify the value for 'ens.ens_thresh' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENS_VLD_THRESH
Specify the value for 'ens.vld_thresh' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_CLIMO
Specify the value for 'ensemble_flag.climo' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_CLIMO_CDP
Specify the value for 'ensemble_flag.climo_cdp' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_FREQUENCY
Specify the value for 'ensemble_flag.frequency' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_LATLON
Specify the value for 'ensemble_flag.latlon' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_MAX
Specify the value for 'ensemble_flag.max' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_MEAN
Specify the value for 'ensemble_flag.mean' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_MIN
Specify the value for 'ensemble_flag.min' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_MINUS
Specify the value for 'ensemble_flag.minus' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_NEP
Specify the value for 'ensemble_flag.nep' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_NMEP
Specify the value for 'ensemble_flag.nmep' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_PLUS
Specify the value for 'ensemble_flag.plus' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_RANGE
Specify the value for 'ensemble_flag.range' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_STDEV
Specify the value for 'ensemble_flag.stdev' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_ENSEMBLE_FLAG_VLD_COUNT
Specify the value for 'ensemble_flag.vld_count' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_INPUT_DATATYPE
Set the file_type entry of the ens dictionary in the MET config file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_INPUT_DIR
Input directory for ensemble files to use with the MET tool gen_ens_prod.

Used by: GenEnsProd

GEN_ENS_PROD_INPUT_FILE_LIST
Specifies an explicit path to a file list file to pass ensembles into gen_ens_prod. If set, *GEN_ENS_PROD_INPUT_TEMPLATE* and *GEN_ENS_PROD_INPUT_DIR* are ignored.

Used by: GenEnsProd

GEN_ENS_PROD_INPUT_TEMPLATE
Template used to specify ensemble input filenames for the MET tool gen_ens_prod.

Used by: GenEnsProd

GEN_ENS_PROD_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: GEN_ENS_PROD_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: GenEnsProd

GEN_ENS_PROD_NBRHD_PROB_SHAPE
Specify the value for 'nbrhd_prob.shape' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_NBRHD_PROB_VLD_THRESH
Specify the value for 'nbrhd_prob.vld_thresh' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_NBRHD_PROB_WIDTH
Specify the value for 'nbrhd_prob.width' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_NC_VAR_STR
Specify the value for 'nc_var_str' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_DX
Specify the value for 'nmep_smooth.gaussian_dx' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_NMEP_SMOOTH_GAUSSIAN_RADIUS
Specify the value for 'nmep_smooth.gaussian_radius' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_NMEP_SMOOTH_METHOD
Specify the value for 'nmep_smooth.type.method' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_NMEP_SMOOTH_SHAPE
Specify the value for 'nmep_smooth.shape' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_NMEP_SMOOTH_VLD_THRESH
Specify the value for 'nmep_smooth.vld_thresh' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_NMEP_SMOOTH_WIDTH
Specify the value for 'nmep_smooth.type.width' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_NORMALIZE
Specify the value for 'normalize' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_OUTPUT_DIR
Specify the output directory where files from the MET gen_ens_prod tool are written.

Used by: GenEnsProd

GEN_ENS_PROD_OUTPUT_TEMPLATE
Specify the output filename template for files written by gen_ens_prod.

Used by: GenEnsProd

GEN_ENS_PROD_REGRID_CENSOR_THRESH
Specify the value for 'regrid.censor_thresh' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_REGRID_CENSOR_VAL
Specify the value for 'regrid.censor_val' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_REGRID_CONVERT
Specify the value for 'regrid.convert' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_REGRID_METHOD
Specify the value for 'regrid.method' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_REGRID_SHAPE
Specify the value for 'regrid.shape' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_REGRID_TO_GRID
Specify the value for 'regrid.to_grid' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_REGRID_VLD_THRESH
Specify the value for 'regrid.vld_thresh' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_ENS_PROD_REGRID_WIDTH
Specify the value for 'regrid.width' in the MET configuration file for GenEnsProd.

Used by: GenEnsProd

GEN_SEQ

Warning: DEPRECATED:

GEN_VX_MASK_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: GenVxMask

GEN_VX_MASK_FILE_WINDOW_BEGIN
Used to control the lower bound of the window around the valid time to determine if a GenVxMask input file should be used for processing. Overrides *FILE_WINDOW_BEGIN*. See 'Use Windows to Find Valid Files' section for more information.

Used by: GenVxMask

GEN_VX_MASK_FILE_WINDOW_END
Used to control the upper bound of the window around the valid time to determine if an GenVxMask input file should be used for processing. Overrides *FILE_WINDOW_BEGIN*. See 'Use Windows to Find Valid Files' section for more information.

Used by: GenVxMask

GEN_VX_MASK_INPUT_DIR
Directory containing input data to GenVxMask. This variable is optional because you can specify a full path to the input files using *GEN_VX_MASK_INPUT_TEMPLATE*.

Used by: GenVxMask

GEN_VX_MASK_INPUT_MASK_DIR
Directory containing mask data used by GenVxMask. This variable is optional because you can specify the full path to the input files using *GEN_VX_MASK_INPUT_MASK_TEMPLATE*.

Used by: GenVxMask

GEN_VX_MASK_INPUT_MASK_TEMPLATE
Filename template of the mask files used by GenVxMask. This can be a list of files or grids separated by commas to apply to the input grid. The wrapper will call GenVxMask once for each item in the list, passing its output to temporary files until the final command, which will write to the file specified by *GEN_VX_MASK_OUTPUT_TEMPLATE* (and optionally *GEN_VX_MASK_OUTPUT_DIR*. The length of this list must be the same length as *GEN_VX_MASK_OPTIONS*. When "-type lat" or "-type lon" is set in *GEN_VX_MASK_OPTIONS*, the corresponding mask template is ignored, but must be set to a placeholder string. See also *GEN_VX_MASK_INPUT_MASK_DIR*.

Used by: GenVxMask

GEN_VX_MASK_INPUT_TEMPLATE
Filename template of the input grid used by GenVxMask. This can be an input filename or a grid definition. See also *GEN_VX_MASK_INPUT_DIR*.

Used by: GenVxMask

GEN_VX_MASK_OPTIONS
Command line arguments to pass to each call of GenVxMask. This can be a list of sets of arguments separated by commas to apply to the input grid. The length of this list must be the same length as *GEN_VX_MASK_INPUT_MASK_TEMPLATE*.

Used by: GenVxMask

GEN_VX_MASK_OUTPUT_DIR
Directory to write output data generated by GenVxMask. This variable is optional because you can specify the full path to the input files using *GEN_VX_MASK_OUTPUT_TEMPLATE*.

Used by: GenVxMask

GEN_VX_MASK_OUTPUT_TEMPLATE
Filename template of the output file generated by GenVxMask. See also *GEN_VX_MASK_OUTPUT_DIR*.

Used by: GenVxMask

GEN_VX_MASK_SKIP_IF_OUTPUT_EXISTS
If True, do not run GenVxMask if output file already exists. Set to False to overwrite files.

Used by: GenVxMask

GFDL_TRACKER_ATCFINFO_ATCFFREQ
Sets the value of &atcfinfo: atcffreq in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_ATCFINFO_ATCFNAME
Sets the value of &atcfinfo: atcfname in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_ATCFINFO_ATCFNUM
Sets the value of &atcfinfo: atcfnum in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_BASE
Path to directory that contains the GFDL Tracker executables such as grbindex.exe and gettrk.exe. In many installations, this is a directory named trk_exec.

Used by: GFDLTracker

GFDL_TRACKER_DATEIN_INP_FILE_SEQ
Sets the value of &datein: inp%file_seq in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_DATEIN_INP_LT_UNITS
Sets the value of &datein: inp%lt_units in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_DATEIN_INP_MODEL
Sets the value of &datein: inp%model in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_DATEIN_INP_MODTYP
Sets the value of &datein: inp%modtyp in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_DATEIN_INP_NESTTYP
Sets the value of &datein: inp%nesttyp in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_FNAMEINFO_ATCFDESCR
Sets the value of &fnameinfo: atcfdescr in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_FNAMEINFO_GMODNAME
Sets the value of &fnameinfo: gmodname in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_FNAMEINFO_RUNDESCR
Sets the value of &fnameinfo: rundescr in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_GRIB_VERSION
Specifies the GRIB version of the input data. Valid values are 1 or 2. This determines which application to use to create the index files (grbindex.exe or grb2index.exe).

Used by: GFDLTracker

GFDL_TRACKER_INPUT_DIR
Directory containing input data to read into GFDLTracker. This is optional as the entire path to the data can be set with *GFDL_TRACKER_INPUT_TEMPLATE*.

Used by: GFDLTracker

GFDL_TRACKER_INPUT_TEMPLATE
Filename template that corresponds to the file naming convention of the input data read into GFDL-Tracker. This can be a full path to a file or a relative path if *GFDL_TRACKER_INPUT_DIR* is set.

Used by: GFDLTracker

GFDL_TRACKER_KEEP_INTERMEDIATE
If True, do not scrub intermediate files created by the tracker. Useful for debugging issues.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_LAT_NAME
Sets the value of &netcdflist: netcdfinfo%lat_name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_LMASKNAME
Sets the value of &netcdflist: netcdfinfo%lmaskname in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_LON_NAME
Sets the value of &netcdflist: netcdfinfo%lon_name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_MSLPNAME
Sets the value of &netcdflist: netcdfinfo%mslpname in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_NETCDF_FILENAME
Sets the value of &netcdflist: netcdfinfo%netcdf_filename in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_NUM_NETCDF_VARS
Sets the value of &netcdflist: netcdfinfo%num_netcdf_vars in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_RV700NAME
Sets the value of &netcdflist: netcdfinfo%rv700name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_RV850NAME
Sets the value of &netcdflist: netcdfinfo%rv850name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_TIME_NAME
Sets the value of &netcdflist: netcdfinfo%time_name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_TIME_UNITS
Sets the value of &netcdflist: netcdfinfo%time_units in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_TMEAN_300_500_NAME
Sets the value of &netcdflist: netcdfinfo%tmean_300_500_name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_U500NAME
Sets the value of &netcdflist: netcdfinfo%u500name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_U700NAME
Sets the value of &netcdflist: netcdfinfo%u700name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_U850NAME
Sets the value of &netcdflist: netcdfinfo%u850name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_USFCNAME
Sets the value of &netcdflist: netcdfinfo%usfcname in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_V500NAME
Sets the value of &netcdflist: netcdfinfo%v500name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_V700NAME
Sets the value of &netcdflist: netcdfinfo%v700name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_V850NAME
Sets the value of &netcdflist: netcdfinfo%v850name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_VSFCNAME
Sets the value of &netcdflist: netcdfinfo%vsfcname in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z200NAME
Sets the value of &netcdflist: netcdfinfo%z200name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z300NAME
Sets the value of &netcdflist: netcdfinfo%z300name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z350NAME
Sets the value of &netcdflist: netcdfinfo%z350name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z400NAME
Sets the value of &netcdflist: netcdfinfo%z400name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z450NAME
Sets the value of &netcdflist: netcdfinfo%z450name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z500NAME
Sets the value of &netcdflist: netcdfinfo%z500name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z550NAME
Sets the value of &netcdflist: netcdfinfo%z550name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z600NAME
Sets the value of &netcdflist: netcdfinfo%z600name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z650NAME
Sets the value of &netcdflist: netcdfinfo%z650name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z700NAME
Sets the value of &netcdflist: netcdfinfo%z700name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z750NAME
Sets the value of &netcdflist: netcdfinfo%z750name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z800NAME
Sets the value of &netcdflist: netcdfinfo%z800name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z850NAME
Sets the value of &netcdflist: netcdfinfo%z850name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NETCDFINFO_Z900NAME
Sets the value of &netcdflist: netcdfinfo%z900name in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_NML_TEMPLATE_FILE
Path to the template NML file that matches the format of the input.nml file that is used by the GFDL Tracker. This file can contain string expressions that are substituted by values read from the METplus configuration variables, so this path likely does not need to be modified.

Used by: GFDLTracker

GFDL_TRACKER_OUTPUT_DIR
Directory to write output data created by GFDLTracker. The tracker application must be run from the directory containing all of the data and configuration files used, so the wrapper will call the application from this directory. Symbolic links for each input file including the TCVitals file will be created in this directory and removed after a successful run. The fort.X files required to run the tracker will be generated in this directory. Also, the input.nml file that is generated from the template NML file (specified by *GFDL_TRACKER_NML_TEMPLATE_FILE*) will be found in this directory.

Used by: GFDLTracker

GFDL_TRACKER_OUTPUT_TEMPLATE
The fort.64 output file that is generated from running the GFDLTracker can be renamed using this variable using filename template syntax to create an output file that contains useful information such as the date.

Used by: GFDLTracker

GFDL_TRACKER_PHASEINFO_PHASEFLAG
Sets the value of &phaseinfo: phaseflag in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_PHASEINFO_PHASESCHEME
Sets the value of &phaseinfo: phasescheme in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_PHASEINFO_WCORE_DEPTH
Sets the value of &phaseinfo: wcore_depth in the template NML file. See

GFDL_TRACKER_NML_TEMPLATE_FILE.

Used by: GFDLTracker

GFDL_TRACKER_STRUCTINFO_IKEFLAG
Sets the value of &structinfo: ikeflag in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_STRUCTINFO_STRUCTFLAG
Sets the value of &structinfo: structflag in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TC_VITALS_INPUT_DIR
Directory containing the TCVitals file that is required to run the GFDLTracker. This is optional as the entire path to the data can be set with *GFDL_TRACKER_TC_VITALS_INPUT_TEMPLATE*.

Used by: GFDLTracker

GFDL_TRACKER_TC_VITALS_INPUT_TEMPLATE
Filename template that corresponds to the file naming convention of the TCVitals file that is required to run the GFDLTracker. This can be a full path to a file or a relative path if *GFDL_TRACKER_TC_VITALS_INPUT_DIR* is set.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_CONTINT
Sets the value of &trackerinfo: trkrinfo%contint in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_ENABLE_TIMING
Sets the value of &trackerinfo: trkrinfo%enable_timing in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_G1_MSLP_PARM_ID
Sets the value of &trackerinfo: trkrinfo%g1_mslp_parm_id in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_TYP
Sets the value of &trackerinfo: trkrinfo%g1_sfcwind_lev_typ in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_G1_SFCWIND_LEV_VAL
Sets the value of &trackerinfo: trkrinfo%g1_sfcwind_lev_val in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_G2_JPDTN
Sets the value of &trackerinfo: trkrinfo%g2_jpdtn in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_G2_MSLP_PARM_ID
Sets the value of &trackerinfo: trkrinfo%g2_mslp_parm_id in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_GRIBVER
Sets the value of &trackerinfo: trkrinfo%gribver in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_GRIDTYPE
Sets the value of &trackerinfo: trkrinfo%gridtype in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_INP_DATA_TYPE
Sets the value of &trackerinfo: trkrinfo%inp_data_type in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_MSLPTHRESH
Sets the value of &trackerinfo: trkrinfo%mslpthresh in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_OUT_VIT
Sets the value of &trackerinfo: trkrinfo%out_vit in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_TYPE
Sets the value of &trackerinfo: trkrinfo%type in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_USE_BACKUP_850_VT_CHECK
Sets the value of &trackerinfo: trkrinfo%use_backup_850_vt_check in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_USE_BACKUP_MSLP_GRAD_CHECK
Sets the value of &trackerinfo: trkrinfo%use_backup_mslp_grad_check in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_USE_LAND_MASK
Sets the value of &trackerinfo: trkrinfo%use_land_mask in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_V850THRESH
Sets the value of &trackerinfo: trkrinfo%v850thresh in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_TRACKERINFO_WANT_OCI
Sets the value of &trackerinfo: trkrinfo%want_oci in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH700
Sets the value of &parmpreflist: user_wants_to_track_gph700 in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_GPH850
Sets the value of &parmpreflist: user_wants_to_track_gph850 in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_MSLP
Sets the value of &parmpreflist: user_wants_to_track_mslp in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200500
Sets the value of &parmpreflist: user_wants_to_track_thick200500 in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK200850
Sets the value of &parmpreflist: user_wants_to_track_thick200850 in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_THICK500850
Sets the value of &parmpreflist: user_wants_to_track_thick500850 in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC700
Sets the value of &parmpreflist: user_wants_to_track_wcirc700 in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRC850
Sets the value of &parmpreflist: user_wants_to_track_wcirc850 in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_WCIRCSFC
Sets the value of &parmpreflist: user_wants_to_track_wcircsfc in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA700
Sets the value of &parmpreflist: user_wants_to_track_zeta700 in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA850
Sets the value of &parmpreflist: user_wants_to_track_zeta850 in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETASFC
Sets the value of &parmpreflist: user_wants_to_track_zetasfc in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_VERBOSE_VERB
Sets the value of &verbose: verb in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_VERBOSE_VERB_G2
Sets the value of &verbose: verb_g2 in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_WAITINFO_PER_FCST_COMMAND
Sets the value of &waitinfo: per_fcst_command in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_WAITINFO_USE_PER_FCST_COMMAND
Sets the value of &waitinfo: use_per_fcst_command in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_WAITINFO_USE_WAITFOR
Sets the value of &waitinfo: use_waitfor in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_WAITINFO_WAIT_MAX_WAIT
Sets the value of &waitinfo: wait_max_wait in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_WAITINFO_WAIT_MIN_AGE
Sets the value of &waitinfo: wait_min_age in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_WAITINFO_WAIT_MIN_SIZE
Sets the value of &waitinfo: wait_min_size in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFDL_TRACKER_WAITINFO_WAIT_SLEEPTIME
Sets the value of &waitinfo: wait_sleeptime in the template NML file. See *GFDL_TRACKER_NML_TEMPLATE_FILE*.

Used by: GFDLTracker

GFS_ANLY_FILE_TMPL

> **Warning:** **DEPRECATED:** Please use *OBS_EXTRACT_TILES_INPUT_TEMPLATE* instead.

GFS_FCST_FILE_TMPL

> **Warning:** **DEPRECATED:** Please use *FCST_EXTRACT_TILES_INPUT_TEMPLATE* instead.

GRID_DIAG_CENSOR_THRESH
Set the censor_thresh entry in the GridDiag MET config file.

Used by: GridDiag

GRID_DIAG_CENSOR_VAL
Set the censor_val entry in the GridDiag MET config file.

Used by: GridDiag

GRID_DIAG_CONFIG_FILE
Path to configuration file read by grid_diag. If unset, parm/met_config/GridDiagConfig_wrapped will be used.

Used by: GridDiag

GRID_DIAG_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: GridDiag

GRID_DIAG_DESC
Specify the value for 'desc' in the MET configuration file for grid_diag.

Used by: GridDiag

GRID_DIAG_INPUT_DATATYPE
Specify the data type of the input directory for files used with the MET grid_diag tool.

Used by: GridDiag

GRID_DIAG_INPUT_DIR
Input directory for files to use with the MET tool grid_diag.

Used by: GridDiag

GRID_DIAG_INPUT_TEMPLATE
Template used to specify input filenames for the MET tool grid_diag. This can be a comma-separated list. If there are more than one template, the number of fields specified must match the number of templates.

Used by: GridDiag

GRID_DIAG_MASK_GRID
Set the mask.grid entry in the GridDiag MET config file.

Used by: GridDiag

GRID_DIAG_MASK_POLY
Set the mask.poly entry in the GridDiag MET config file.

Used by: GridDiag

GRID_DIAG_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: GRID_DIAG_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: GridDiag

GRID_DIAG_OUTPUT_DIR
Output directory for write files with the MET tool grid_diag.

Used by: GridDiag

GRID_DIAG_OUTPUT_TEMPLATE
Template used to specify output filenames created by MET tool grid_diag.

Used by: GridDiag

GRID_DIAG_REGRID_CENSOR_THRESH
Specify the value for 'regrid.censor_thresh' in the MET configuration file for GridDiag.

Used by: GridDiag

GRID_DIAG_REGRID_CENSOR_VAL
Specify the value for 'regrid.censor_val' in the MET configuration file for GridDiag.

Used by: GridDiag

GRID_DIAG_REGRID_CONVERT
Specify the value for 'regrid.convert' in the MET configuration file for GridDiag.

Used by: GridDiag

GRID_DIAG_REGRID_METHOD
Specify the value for 'regrid.method' in the MET configuration file for grid_diag.

Used by: GridDiag

GRID_DIAG_REGRID_SHAPE
Specify the value for 'regrid.shape' in the MET configuration file for grid_diag.

Used by: GridDiag

GRID_DIAG_REGRID_TO_GRID
Specify the value for 'regrid.to_grid' in the MET configuration file for grid_diag.

Used by: GridDiag

GRID_DIAG_REGRID_VLD_THRESH
Specify the value for 'regrid.vld_thresh' in the MET configuration file for grid_diag.

Used by: GridDiag

GRID_DIAG_REGRID_WIDTH
Specify the value for 'regrid.width' in the MET configuration file for grid_diag.

Used by: GridDiag

GRID_DIAG_RUNTIME_FREQ
Frequency to run Grid-Diag. See *Runtime Frequency* (page 70) for more information.

Used by: GridDiag

GRID_DIAG_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: GridDiag

GRID_DIAG_VERIFICATION_MASK_TEMPLATE
Template used to specify the verification mask filename for the MET tool grid_diag. Supports a list of filenames.

Used by: GridDiag

GRID_STAT_CENSOR_THRESH
Specify the value for 'censor_thresh' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CENSOR_VAL
Specify the value for 'censor_val' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_CDF_BINS
Specify the value for 'climo_cdf.cdf_bins' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_CDF_CDF_BINS
See *GRID_STAT_CLIMO_CDF_BINS*

GRID_STAT_CLIMO_CDF_CENTER_BINS
Specify the value for 'climo_cdf.center_bins' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_CDF_DIRECT_PROB
Specify the value for 'climo_cdf.direct_prob' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_CDF_WRITE_BINS
Specify the value for 'climo_cdf.write_bins' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_DAY_INTERVAL
Specify the value for 'climo_mean.day_interval' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_FIELD
See: *<TOOL-NAME>_CLIMO_MEAN_FIELD*

Used by: GridStat

GRID_STAT_CLIMO_MEAN_FILE_NAME
Specify the value for 'climo_mean.file_name' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_HOUR_INTERVAL
Specify the value for 'climo_mean.hour_interval' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *GRID_STAT_CLIMO_MEAN_FILE_NAME*.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *GRID_STAT_CLIMO_MEAN_FILE_NAME*.

GRID_STAT_CLIMO_MEAN_MATCH_MONTH
Specify the value for 'climo_mean.match_month' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_REGRID_METHOD
Specify the value for 'climo_mean.regrid.method' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_REGRID_SHAPE
Specify the value for 'climo_mean.regrid.shape' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_REGRID_VLD_THRESH
Specify the value for 'climo_mean.regrid.vld_thresh' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_REGRID_WIDTH
Specify the value for 'climo_mean.regrid.width' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_TIME_INTERP_METHOD
Specify the value for 'climo_mean.time_interp_method' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_USE_FCST
If set to True, use the field array from the fcst dictionary for the climo_mean fields for GridStat. Sets "climo_mean = fcst;" in the wrapped MET config file. Only used if *GRID_STAT_CLIMO_MEAN_FIELD* is unset. See also *GRID_STAT_CLIMO_MEAN_USE_OBS*.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_USE_OBS
If set to True, use the field array from the obs dictionary for the climo_mean fields for GridStat. Sets "climo_mean = obs;" in the wrapped MET config file. Only used if *GRID_STAT_CLIMO_MEAN_FIELD* is unset. See also *GRID_STAT_CLIMO_MEAN_USE_FCST*.

Used by: GridStat

GRID_STAT_CLIMO_MEAN_VAR<n>_LEVELS
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_LEVELS*

Used by: GridStat

GRID_STAT_CLIMO_MEAN_VAR<n>_NAME
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_NAME*

Used by: GridStat

GRID_STAT_CLIMO_MEAN_VAR<n>_OPTIONS
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_OPTIONS*

Used by: GridStat

GRID_STAT_CLIMO_STDEV_DAY_INTERVAL
Specify the value for 'climo_stdev.day_interval' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_FIELD
Specify the value for 'climo_stdev.field' in the MET configuration file for GridStat. The value set here must include the proper formatting that is expected in MET configuration file for specifying field information. Example: {name="TMP"; level="(,)";} To set the field information un-formatted, use the *GRID_STAT_CLIMO_STDEV_VAR<n>_NAME*, *GRID_STAT_CLIMO_STDEV_VAR<n>_LEVELS*, and *GRID_STAT_CLIMO_STDEV_VAR<n>_OPTIONS* variables.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_FILE_NAME
Specify the value for 'climo_stdev.file_name' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_HOUR_INTERVAL
Specify the value for 'climo_stdev.hour_interval' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_INPUT_DIR

> **Warning: DEPRECATED:** Please use *GRID_STAT_CLIMO_STDEV_FILE_NAME*.

GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE

> **Warning: DEPRECATED:** Please use *GRID_STAT_CLIMO_STDEV_FILE_NAME*.

GRID_STAT_CLIMO_STDEV_MATCH_MONTH
Specify the value for 'climo_stdev.match_month' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_REGRID_METHOD
Specify the value for 'climo_stdev.regrid.method' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_REGRID_SHAPE
Specify the value for 'climo_stdev.regrid.shape' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_REGRID_VLD_THRESH
Specify the value for 'climo_stdev.regrid.vld_thresh' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_REGRID_WIDTH
Specify the value for 'climo_stdev.regrid.width' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_TIME_INTERP_METHOD
Specify the value for 'climo_stdev.time_interp_method' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_USE_FCST
If set to True, use the field array from the fcst dictionary for the climo_stdev fields for GridStat. Sets "climo_stdev = fcst;" in the wrapped MET config file. Only used if *GRID_STAT_CLIMO_STDEV_FIELD* is unset. See also *GRID_STAT_CLIMO_STDEV_USE_OBS*.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_USE_OBS
If set to True, use the field array from the obs dictionary for the climo_stdev fields for GridStat. Sets "climo_stdev = obs;" in the wrapped MET config file. Only used if *GRID_STAT_CLIMO_STDEV_FIELD* is unset. See also *GRID_STAT_CLIMO_STDEV_USE_FCST*.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_VAR<n>_LEVELS
Specify the level of the nth field for 'climo_stdev.field' in the MET configuration file for GridStat. If any fields are set using this variable, then *GRID_STAT_CLIMO_STDEV_FIELD* will be ignored. See also *GRID_STAT_CLIMO_STDEV_VAR<n>_NAME* and *GRID_STAT_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_VAR<n>_NAME
Specify the name of the nth field for 'climo_stdev.field' in the MET configuration file for GridStat. If any fields are set using this variable, then *GRID_STAT_CLIMO_STDEV_FIELD* will be ignored. See also *GRID_STAT_CLIMO_STDEV_VAR<n>_LEVELS* and *GRID_STAT_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: GridStat

GRID_STAT_CLIMO_STDEV_VAR<n>_OPTIONS
Specify the extra options of the nth field for 'climo_stdev.field' in the MET configuration file for GridStat. If any fields are set using this variable, then *GRID_STAT_CLIMO_STDEV_FIELD* will be ignored. See also *GRID_STAT_CLIMO_STDEV_VAR<n>_NAME* and *GRID_STAT_CLIMO_STDEV_VAR<n>_LEVELS*.

Used by: GridStat

GRID_STAT_CONFIG

> **Warning:** **DEPRECATED:** Please use *GRID_STAT_CONFIG_FILE* instead.

GRID_STAT_CONFIG_FILE
Path to configuration file read by grid_stat. If unset, parm/met_config/GridStatConfig_wrapped will be used.

Used by: GridStat

GRID_STAT_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: GridStat

GRID_STAT_DESC
Specify the value for 'desc' in the MET configuration file for grid_stat.

Used by: GridStat

GRID_STAT_DISTANCE_MAP_BADDELEY_MAX_DIST
Specify the value for 'distance_map.baddeley_max_dist' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_DISTANCE_MAP_BADDELEY_P
Specify the value for 'distance_map.baddeley_p' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_DISTANCE_MAP_BETA_VALUE_N
Specify the value for 'distance_map.beta_value(n)' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_DISTANCE_MAP_FOM_ALPHA
Specify the value for 'distance_map.fom_alpha' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_DISTANCE_MAP_ZHU_WEIGHT
Specify the value for 'distance_map.zhu_weight' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_FOURIER_WAVE_1D_BEG
Specify the value for 'fourier.wave_1d_beg' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_FOURIER_WAVE_1D_END
Specify the value for 'fourier.wave_1d_end' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_GRID_WEIGHT_FLAG
Specify the value for 'grid_weight_flag' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_HSS_EC_VALUE
Specify the value for 'hss_ec_value' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_INTERP_FIELD
Specify the value for 'interp.field' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_INTERP_SHAPE
Specify the value for 'interp.shape' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_INTERP_TYPE_METHOD
Specify the value for 'interp.type.method' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_INTERP_TYPE_WIDTH
Specify the value for 'interp.type.width' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_INTERP_VLD_THRESH
Specify the value for 'interp.vld_thresh' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_MASK_GRID
Specify the value for 'mask.grid' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_MASK_POLY
Specify the value for 'mask.poly' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_MET_CONFIG_OVERRIDES

Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: GRID_STAT_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK

Specify the value for 'nc_pairs_flag.apply_mask' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_CLIMO

Specify the value for 'nc_pairs_flag.climo' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_CLIMO_CDP

Specify the value for 'nc_pairs_flag.climo_cdp' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_DIFF

Specify the value for 'nc_pairs_flag.diff' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_DISTANCE_MAP

Specify the value for 'nc_pairs_flag.distance_map' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_FOURIER
Specify the value for 'nc_pairs_flag.fourier' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_GRADIENT
Specify the value for 'nc_pairs_flag.gradient' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_LATLON
Specify the value for 'nc_pairs_flag.latlon' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_NBRHD
Specify the value for 'nc_pairs_flag.nbrhd' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_RAW
Specify the value for 'nc_pairs_flag.raw' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_SEEPS
Specify the value for 'nc_pairs_flag.seeps' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_FLAG_WEIGHT
Specify the value for 'nc_pairs_flag.weight' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NC_PAIRS_VAR_NAME
Specify the value for 'nc_pairs_var_name' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_NEIGHBORHOOD_COV_THRESH
Sets the neighborhood cov_thresh list used by GridStat. See MET User's Guide for more information.

Used by: GridStat

GRID_STAT_NEIGHBORHOOD_SHAPE
Sets the neighborhood shape used by GridStat. See MET User's Guide for more information.

Used by: GridStat

GRID_STAT_NEIGHBORHOOD_WIDTH
Sets the neighborhood width used by GridStat. See MET User's Guide for more information.

Used by: GridStat

GRID_STAT_ONCE_PER_FIELD
True/False. If True, grid_stat will run once to process all name/level/threshold combinations specified. If False, it will run once for each name/level. Some cases require this to be set to False, for example processing probablistic forecasts or precipitation accumulations.

Used by: GridStat

GRID_STAT_OUT_DIR

Warning: **DEPRECATED:** Please use *GRID_STAT_OUTPUT_DIR* instead.

GRID_STAT_OUTPUT_DIR
Specify the output directory where files from the MET grid_stat tool are written.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_CNT
Specify the value for 'output_flag.cnt' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_CTC
Specify the value for 'output_flag.ctc' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_CTS
Specify the value for 'output_flag.cts' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_DMAP
Specify the value for 'output_flag.dmap' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_ECLV
Specify the value for 'output_flag.eclv' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_FHO
Specify the value for 'output_flag.fho' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_GRAD
Specify the value for 'output_flag.grad' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_MCTC
Specify the value for 'output_flag.mctc' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_MCTS
Specify the value for 'output_flag.mcts' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_NBRCNT
Specify the value for 'output_flag.nbrcnt' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_NBRCTC
Specify the value for 'output_flag.nbrctc' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_NBRCTS
Specify the value for 'output_flag.nbrcts' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_PCT
Specify the value for 'output_flag.pct' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_PJC
Specify the value for 'output_flag.pjc' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_PRC
Specify the value for 'output_flag.prc' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_PSTD
Specify the value for 'output_flag.pstd' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_SAL1L2
Specify the value for 'output_flag.sal1l2' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_SEEPS
Specify the value for 'output_flag.seeps' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_SL1L2
Specify the value for 'output_flag.sl1l2' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_VAL1L2
Specify the value for 'output_flag.val1l2' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_VCNT
Specify the value for 'output_flag.vcnt' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_FLAG_VL1L2
Specify the value for 'output_flag.vl1l2' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_OUTPUT_PREFIX
String to pass to the MET config file to prepend text to the output filenames.

Used by: GridStat

GRID_STAT_OUTPUT_TEMPLATE
Sets the subdirectories below *GRID_STAT_OUTPUT_DIR* using a template to allow run time information. If LOOP_BY = VALID, default value is valid time YYYYMMDDHHMM/grid_stat. If LOOP_BY = INIT, default value is init time YYYYMMDDHHMM/grid_stat.

Used by: GridStat

GRID_STAT_REGRID_CENSOR_THRESH
Specify the value for 'regrid.censor_thresh' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_REGRID_CENSOR_VAL
Specify the value for 'regrid.censor_val' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_REGRID_CONVERT
Specify the value for 'regrid.convert' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_REGRID_METHOD
Specify the value for 'regrid.method' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_REGRID_SHAPE
Specify the value for 'regrid.shape' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_REGRID_TO_GRID
Used to set the regrid dictionary item 'to_grid' in the MET GridStat config file. See the MET User's Guide for more information.

Used by: GridStat

GRID_STAT_REGRID_VLD_THRESH
Specify the value for 'regrid.vld_thresh' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_REGRID_WIDTH
Specify the value for 'regrid.width' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_SEEPS_P1_THRESH
Specify the value for 'seeps_p1_thresh' in the MET configuration file for GridStat.

Used by: GridStat

GRID_STAT_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: GridStat

GRID_STAT_VERIFICATION_MASK_TEMPLATE
Template used to specify the verification mask filename for the MET tool grid_stat. Now supports a list of filenames.

Used by: GridStat

GROUP_LIST_ITEMS
Names of the lists in the METplus .conf file to treat the items in those lists as a group.

Used by: StatAnalysis

HFIP_BASELINE

> **Warning:** **DEPRECATED:** Please use *TCMPR_PLOTTER_HFIP_BASELINE* instead.

INIT_BEG
Specify the beginning initialization time to be used in the analysis. Format can be controlled by *INIT_TIME_FMT*. See *Looping by Initialization Time* (page 48) for more information.

Used by: All

INIT_END
Specify the ending initialization time to be used in the analysis. Format can be controlled by *INIT_TIME_FMT*. See *Looping by Initialization Time* (page 48) for more information.

Used by: All

INIT_EXCLUDE

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_INIT_EXCLUDE* instead.

INIT_HOUR_BEG

> **Warning:** **DEPRECATED:** Please use *FCST_INIT_HOUR_LIST* or *OBS_INIT_HOUR_LIST* instead.

INIT_HOUR_END

> **Warning: DEPRECATED:** Please use *FCST_INIT_HOUR_LIST* or *OBS_INIT_HOUR_LIST* instead.

INIT_HOUR_INCREMENT

> **Warning: DEPRECATED:** Please use *FCST_INIT_HOUR_LIST* or *OBS_INIT_HOUR_LIST* instead.

INIT_HOUR_METHOD

> **Warning: DEPRECATED:** No longer used.

INIT_INCLUDE

> **Warning: DEPRECATED:** Please use *TC_PAIRS_INIT_INCLUDE* instead.

INIT_INCREMENT

Control the increment or stride to use when stepping between forecast initializations. Units are seconds. See *Looping by Initialization Time* (page 48) for more information. Units are assumed to be seconds unless specified with Y, m, d, H, M, or S.

Used by: All

INIT_LIST

List of initialization times to process. This variable is used when intervals between run times are irregular. It is only read if *LOOP_BY* = INIT. If it is set, then *INIT_BEG*, *INIT_END*, and *INIT_INCREMENT* are ignored. All values in the list must match the format of *INIT_TIME_FMT* or they will be skipped.

Used by: All

INIT_SEQ

Specify a list of initialization hours that are used to build a sequence of forecast lead times to include in the analysis. Used only when looping by valid time (LOOP_BY = VALID). Comma separated list format, e.g.:0, 6, 12 See *Looping over Forecast Leads* (page 50) for more information.

Used by: EnsembleStat, GridStat, MODE, MTD, PB2NC, PCPCombine, PointStat, RegridDataPlane, SeriesAnalysis

INIT_TIME_FMT
Specify a formatting string to use for *INIT_BEG* and *INIT_END*. See *Looping by Initialization Time* (page 48) for more information.

Used by: All

INPUT_BASE
Provide a path to the top level output directory for METplus. It is required and must be set correctly to run any of the use cases. This can be the location of sample input data to run use cases found in the METplus repository. Each of the sample data tarballs attached to the METplus release should be untarred in this directory. If done correctly, this directory should contain a directory named 'met_test' and a directory named 'model_applications.'

Used by: All

INTERP

Warning: DEPRECATED: Please use *INTERP_MTHD_LIST* instead.

INTERP_MTHD_LIST
Specify the values of the INTERP_MTHD column in the MET .stat file to use; specify the interpolation used to create the MET .stat files.

Groups of values can be looped over by setting INTERP_MTHD_LIST<n> and adding INTERP_MTHD_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

INTERP_PNTS_LIST
Specify the values of the INTERP_PNTS column in the MET .stat file to use; corresponds to the interpolation in the MET .stat files.

Groups of values can be looped over by setting INTERP_PNTS_LIST<n> and adding INTERP_PNTS_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

INTERP_PTS

> **Warning: DEPRECATED:** Please use *INTERP_PNTS_LIST* instead.

INTERVAL_TIME
Define the interval time in hours (HH) to be used by the MET pb2nc tool.

Used by: PB2NC

IODA2NC_CONFIG_FILE
Path to wrapped MET configuration file read by ioda2nc. If unset, {PARM_BASE}/met_config/IODA2NCConfig_wrapped will be used.

Used by: IODA2NC

IODA2NC_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: IODA2NC

IODA2NC_ELEVATION_RANGE_BEG
Specify the value for 'elevation_range.beg' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_ELEVATION_RANGE_END
Specify the value for 'elevation_range.end' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_FILE_WINDOW_BEG
Used to control the lower bound of the window around the valid time to determine if an IODA2NC input file should be used for processing. Overrides *OBS_FILE_WINDOW_BEGIN*. See 'Use Windows to Find Valid Files' section for more information.

Used by: IODA2NC

IODA2NC_FILE_WINDOW_END
Used to control the upper bound of the window around the valid time to determine if an IODA2NC input file should be used for processing. Overrides *OBS_FILE_WINDOW_END*. See 'Use Windows to Find Valid Files' section for more information.

Used by: IODA2NC

IODA2NC_INPUT_DIR
Directory containing input data to IODA2NC. This variable is optional because you can specify the full path to the input files using *IODA2NC_INPUT_TEMPLATE*.

Used by: IODA2NC

IODA2NC_INPUT_TEMPLATE
Filename template of the input file used by IODA2NC. See also *IODA2NC_INPUT_DIR*.

Used by: IODA2NC

IODA2NC_LEVEL_RANGE_BEG
Specify the value for 'level_range.beg' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_LEVEL_RANGE_END
Specify the value for 'level_range.end' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_MASK_GRID
Specify the value for 'mask.grid' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_MASK_POLY
Specify the value for 'mask.poly' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_MESSAGE_TYPE
Specify the value for 'message_type' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_MESSAGE_TYPE_GROUP_MAP
Specify the value for 'message_type_group_map' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_MESSAGE_TYPE_MAP
Specify the value for 'message_type_map' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: IODA2NC_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: IODA2NC

IODA2NC_METADATA_MAP
Specify the value for 'metadata_map' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_MISSING_THRESH
Specify the value for 'missing_thresh' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_NMSG
Used to set the command line argument -nmsg for ioda2nc.

Used by: IODA2NC

IODA2NC_OBS_NAME_MAP
Specify the value for 'obs_name_map' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_OBS_VAR
Specify the value for 'obs_var' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_OBS_WINDOW_BEG
Specify the value for 'obs_window.beg' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_OBS_WINDOW_END
Specify the value for 'obs_window.end' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_OUTPUT_DIR
Directory to write output data generated by IODA2NC. This variable is optional because you can specify the full path to the output files using *IODA2NC_OUTPUT_TEMPLATE*.

Used by: IODA2NC

IODA2NC_OUTPUT_TEMPLATE
Filename template of the output file generated by IODA2NC. See also *IODA2NC_OUTPUT_DIR*.

Used by: IODA2NC

IODA2NC_QUALITY_MARK_THRESH
Specify the value for 'quality_mark_thresh' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_SKIP_IF_OUTPUT_EXISTS
If True, do not run IODA2NC if output file already exists. Set to False to overwrite files.

Used by: IODA2NC

IODA2NC_STATION_ID
Specify the value for 'station_id' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_TIME_SUMMARY_BEG
Specify the value for 'time_summary.beg' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_TIME_SUMMARY_END
Specify the value for 'time_summary.end' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_TIME_SUMMARY_FLAG
Specify the value for 'time_summary.flag' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_TIME_SUMMARY_GRIB_CODE
Specify the value for 'time_summary.grib_code' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_TIME_SUMMARY_OBS_VAR
Specify the value for 'time_summary.obs_var' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_TIME_SUMMARY_RAW_DATA
Specify the value for 'time_summary.raw_data' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_TIME_SUMMARY_STEP
Specify the value for 'time_summary.step' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_TIME_SUMMARY_TYPE
Specify the value for 'time_summary.type' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_TIME_SUMMARY_VLD_FREQ
Specify the value for 'time_summary.vld_freq' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_TIME_SUMMARY_VLD_THRESH
Specify the value for 'time_summary.vld_thresh' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_TIME_SUMMARY_WIDTH
Specify the value for 'time_summary.width' in the MET configuration file for IODA2NC.

Used by: IODA2NC

IODA2NC_VALID_BEG
Used to set the command line argument -valid_beg that controls the lower bound of valid times of data to use. Filename template notation can be used, i.e. {valid?fmt=%Y%m%d_%H%M%S}

Used by: IODA2NC

IODA2NC_VALID_END
Used to set the command line argument -valid_end that controls the upper bound of valid times of data to use. Filename template notation can be used, i.e. {valid?fmt=%Y%m%d_%H%M%S?shift=1d} (valid time shifted forward one day)

Used by: IODA2NC

JOB_ARGS

Warning: **DEPRECATED:** Please use *STAT_ANALYSIS_JOB_ARGS* instead.

LAT_ADJ

Warning: **DEPRECATED:** Please use *EXTRACT_TILES_LAT_ADJ* instead.

LEAD

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_LEAD* instead.

LEAD_LIST

Warning: **DEPRECATED:** Please use *FCST_LEAD_LIST* instead.

LEAD_SEQ
Specify the sequence of forecast lead times to include in the analysis. Comma separated list format, e.g.:0, 6, 12. See *Looping over Forecast Leads* (page 50) for more information. Units are assumed to be hours unless specified with Y, m, d, H, M, or S.

Used by: All

LEAD_SEQ_<n>
Specify the sequence of forecast lead times to include in the analysis. Comma separated list format, e.g.:0, 6, 12. <n> corresponds to the bin in which the user wishes to aggregate series by lead results.

Used by: SeriesAnalysis

LEAD_SEQ_<n>_LABEL
Required when SERIES_BY_LEAD_GROUP_FCSTS=True. Specify the label of the corresponding bin of series by lead results.

Used by: SeriesAnalysis

LEAD_SEQ_MAX
Maximum forecast lead to be processed. Used primarily with *INIT_SEQ* but also affects *LEAD_SEQ*. See *Looping over Forecast Leads* (page 50) for more information. Units are assumed to be hours unless specified with Y, m, d, H, M, or S.

Used by: All

LEAD_SEQ_MIN
Minimum forecast lead to be processed. Used primarily with INIT_SEQ but also affects LEAD_SEQ. See *Looping over Forecast Leads* (page 50) for more information. Units are assumed to be hours unless specified with Y, m, d, H, M, or S.

Used by: All

LEGEND

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_LEGEND* instead.

LINE_TYPE

Warning: **DEPRECATED:** Please use *LINE_TYPE_LIST* instead.

LINE_TYPE_LIST
Specify the MET STAT line types to be considered.

Groups of values can be looped over by setting LINE_TYPE_LIST<n> and adding LINE_TYPE_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis, TCMPRPlotter

LOG_ASCII2NC_VERBOSITY
Overrides the log verbosity for ASCII2NC only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: ASCII2NC

LOG_DIR
Specify the directory where log files from MET and METplus should be written.

Used by: All

LOG_ENSEMBLE_STAT_VERBOSITY
Overrides the log verbosity for EnsembleStat only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: EnsembleStat

LOG_GEN_ENS_PROD_VERBOSITY
Overrides the log verbosity for GenEnsProd only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: GenEnsProd

LOG_GEN_VX_MASK_VERBOSITY
Overrides the log verbosity for GenVxMask only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: GenVxMask

LOG_GRID_DIAG_VERBOSITY
Overrides the log verbosity for GridDiag only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: GridDiag

LOG_GRID_STAT_VERBOSITY
Overrides the log verbosity for GridStat only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: GridStat

LOG_IODA2NC_VERBOSITY
Overrides the log verbosity for IODA2NC only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: IODA2NC

LOG_LEVEL
Specify the level of logging for the METplus wrapper output to the *LOG_METPLUS* log file. Log level of the applications that are called by the wrappers are controlled with *LOG_MET_VERBOSITY*. Default log level is INFO. Set to DEBUG to see additional log output. Log level for screen output can be set with *LOG_LEVEL_TERMINAL*.

Options (ordered LEAST verbose to MOST verbose): CRITICAL, ERROR, WARNING, INFO, DEBUG, NOTSET

Used by: All

LOG_LEVEL_TERMINAL
Specify the level of logging for terminal screen output. See *LOG_LEVEL* for more information.

Used by: All

LOG_LINE_DATE_FORMAT
Defines the formatting of the date in the METplus log output. See *LOG_LINE_FORMAT*.

Used by: All

LOG_LINE_FORMAT
Defines the formatting of each METplus log output line. For more information on acceptable values, see the Python documentation for LogRecord: https://docs.python.org/3/library/logging.html#logging.LogRecord

Used by: All

LOG_MET_OUTPUT_TO_METPLUS
Control whether logging output from each executable is sent to the METplus log file or individual log files.

Used by: All

LOG_MET_VERBOSITY
Control the verbosity of the logging from the MET tools.0 = Least amount of logging (lowest verbosity)5 = Most amount of logging (highest verbosity)

Used by: All

LOG_METPLUS
Path to the METplus log file. Control the timestamp appended to the filename with *LOG_TIMESTAMP_TEMPLATE*. Set this variable to an empty string to turn off all logging.

Used by: All

LOG_MODE_VERBOSITY
Overrides the log verbosity for MODE only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: MODE

LOG_MTD_VERBOSITY
Overrides the log verbosity for MTD only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: MTD

LOG_PB2NC_VERBOSITY
Overrides the log verbosity for PB2NC only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: PB2NC

LOG_PCP_COMBINE_VERBOSITY
Overrides the log verbosity for PCPCombine only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: PCPCombine

LOG_PLOT_DATA_PLANE_VERBOSITY
Overrides the log verbosity for PlotDataPlane only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*

Used by: PlotDataPlane

LOG_PLOT_POINT_OBS_VERBOSITY
Overrides the log verbosity for plot_point_obs only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: PlotPointObs

LOG_POINT_STAT_VERBOSITY
Overrides the log verbosity for PointStat only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: PointStat

LOG_REGRID_DATA_PLANE_VERBOSITY
Overrides the log verbosity for RegridDataPlane only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: RegridDataPlane

LOG_SERIES_ANALYSIS_VERBOSITY
Overrides the log verbosity for SeriesAnalysis only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: SeriesAnalysis

LOG_STAT_ANALYSIS_VERBOSITY
Overrides the log verbosity for StatAnalysis only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: StatAnalysis

LOG_TC_DIAG_VERBOSITY
Overrides the log verbosity for TCDiag only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: TCDiag

LOG_TC_GEN_VERBOSITY
Overrides the log verbosity for TCGen only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: TCGen

LOG_TC_PAIRS_VERBOSITY
Overrides the log verbosity for TCPairs only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: TCPairs

LOG_TC_RMW_VERBOSITY
Overrides the log verbosity for TCRMW only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: TCRMW

LOG_TC_STAT_VERBOSITY
Overrides the log verbosity for TCStat only. If not set, the verbosity level is controlled by *LOG_MET_VERBOSITY*.

Used by: TCStat

LOG_TIMESTAMP
Automatically set by METplus based on the values set for *LOG_TIMESTAMP_TEMPLATE* and *LOG_TIMESTAMP_USE_DATATIME*. Setting this in a configuration file will have no effect.

Used by: All

LOG_TIMESTAMP_TEMPLATE
Set the timestamp template used to set *LOG_TIMESTAMP*. Use only Python strftime directives, e.g. %Y%m%d for YYYYMMDD. See also *LOG_TIMESTAMP_USE_DATATIME*.

Used by: All

LOG_TIMESTAMP_USE_DATATIME
True/False. Determines which time to use for the log filenames. If True, use *INIT_BEG* if LOOP_BY is INIT or *VALID_BEG* if LOOP_BY is VALID. If False, use current time.

Used by: All

LON_ADJ

Warning: **DEPRECATED:** Please use *EXTRACT_TILES_LON_ADJ* instead.

LOOP_BY
Control whether the analysis is processed across valid or initialization times. See section *LOOP_BY* (page 47) for more information.

Used by: All

LOOP_BY_INIT

> **Warning:** **DEPRECATED:** Please use *LOOP_BY* instead.

LOOP_LIST_ITEMS
Names of the lists in the METplus .conf file to treat the items in those lists individually.

Used by: StatAnalysis

LOOP_ORDER

> **Warning:** **DEPRECATED:** This previously controlled the looping order for METplus. This was removed in v5.0.0. The wrappers will always execute the logic that was previously run when LOOP_ORDER = processes, which runs each item in the *PROCESS_LIST* for all times specified, then repeat for the next item in the *PROCESS_LIST*.

Used by: All

MET_BASE

> **Warning:** **DEPRECATED:** Do not set.

MET_BIN

> **Warning:** **DEPRECATED:** Please use *MET_INSTALL_DIR* instead.

MET_BIN_DIR
The directory of the MET executables. Used to get the full path of the MET executable when calling from METplus Wrappers. When using the –bindir option in configuring MET, set MET_BIN_DIR to the same location. MET_BIN_DIR will be set to {MET_INSTALL_DIR}/bin. Users can unset MET_BIN_DIR or set it to an empty string if the MET tools are found in the user's path, e.g. when using module loads. | *Used by:* All

MET_BUILD_BASE
The base directory of the MET install. Only needed if using MET version 6.0

Used by: TCMPRPlotter

MET_DATA_DB_DIR
Set this the location of the dtcenter/METdataio repository.

Used by: METdbLoad

MET_DB_LOAD_INPUT_TEMPLATE
Path to a directory containing .stat or .tcst file that will be loaded into METviewer. This can be a single directory or a list of directories. The paths can include filename template tags that correspond to each run time. The wrapper will traverse through each sub directory under the directories listed here and add any directory that contains any files that end with .stat or .tcst to the XML file that is passed into the met_db_load.py script.

Used by: METdbLoad

MET_DB_LOAD_MV_APPLY_INDEXES
Set the <load_spec><apply_indexes> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_DATABASE
Set the <load_spec><connection><database> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_DROP_INDEXES
Set the <load_spec><drop_indexes> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_GROUP
Set the <load_spec><group> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_HOST
Set the <load_spec><connection><host> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_INSERT_SIZE
Set the <load_spec><insert_size> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_LOAD_MODE
Set the <load_spec><load_mode> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_LOAD_MPR
Set the <load_spec><load_mpr> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_LOAD_MTD
Set the <load_spec><load_mtd> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_LOAD_STAT
Set the <load_spec><load_stat> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_MODE_HEADER_DB_CHECK
Set the <load_spec><mode_header_db_check> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_PASSWORD
Set the <load_spec><connection><password> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_USER
Set the <load_spec><connection><user> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_MV_VERBOSE
Set the <load_spec><verbose> value in the METdbLoad XML template file.

Used by: METdbLoad

MET_DB_LOAD_REMOVE_TMP_XML
If set to False, then the temporary XML file with substituted values will not be removed after the use case finishes. This is used for debugging purposes only. The temporary XML file may contain sensitive information like database credentials so it is recommended to remove the temporary file after each run.

Used by: METdbLoad

MET_DB_LOAD_RUNTIME_FREQ
Frequency to run Grid-Diag. See *Runtime Frequency* (page 70) for more information.

Used by: GridDiag

MET_DB_LOAD_XML_FILE
Template XML file that is used to load data into METviewer using the met_db_load.py script. Values from the METplus configuration file are substituted into this file before passing it to the script. The default value can be used to run unless the template doesn't fit the needs of the use case.

Used by: METdbLoad

MET_INSTALL_DIR
The base directory of the MET install. To be defined when using MET version 6.1 and beyond. Used to get the full path of the MET executable and the share directory when calling from METplus Wrappers.

Used by: All

METPLUS_BASE
This variable will automatically be set by METplus when it is started. It will be set to the location of METplus that is currently being run. Setting this variable in a config file will have no effect and will report a warning that it is being overridden.

Used by: All

METPLUS_CONF
Path to the final METplus configuration file. This file will contain every configuration option and value used when METplus was run, including any default values that were used. By default the filename includes the *LOG_TIMESTAMP* so the final conf file and the corresponding log file can be reviewed.

Used by: All

MISSING_VAL

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_MISSING_VAL*.

MISSING_VAL_TO_REPLACE

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_MISSING_VAL_TO_REPLACE*.

MODE_CONFIG

> **Warning:** **DEPRECATED:** Please use *MODE_CONFIG_FILE* instead. Path to mode configuration file.

MODE_CONFIG_FILE
Path to configuration file read by mode. If unset, parm/met_config/MODEConfig_wrapped will be used.

Used by: MODE

MODE_CONV_RADIUS
Comma separated list of convolution radius values used by mode for both forecast and observation

fields. Has the same behavior as setting *FCST_MODE_CONV_RADIUS* and *OBS_MODE_CONV_RADIUS* to the same value.

Used by: MODE

MODE_CONV_THRESH
Comma separated list of convolution threshold values used by mode for both forecast and observation fields. Has the same behavior as setting *FCST_MODE_CONV_THRESH* and *OBS_MODE_CONV_THRESH* to the same value.

Used by: MODE

MODE_CT_STATS_FLAG
Specify the value for 'ct_stats_flag' in the MET configuration file for MODE.

Used by: MODE

MODE_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: MODE

MODE_DESC
Specify the value for 'desc' in the MET configuration file for MODE.

Used by: MODE

MODE_FCST_CENSOR_THRESH
Specify the value for 'fcst.censor_thresh' in the MET configuration file for MODE.

Used by: MODE

MODE_FCST_CENSOR_VAL
Specify the value for 'fcst.censor_val' in the MET configuration file for MODE.

Used by: MODE

MODE_FCST_CONV_RADIUS
Comma separated list of convolution radius values used by mode for forecast fields.

Used by: MODE

MODE_FCST_CONV_THRESH
Comma separated list of convolution threshold values used by mode for forecast fields.

Used by: MODE

MODE_FCST_FILE_TYPE
Specify the value for 'fcst.file_type' in the MET configuration file for MODE.

Used by: MODE

MODE_FCST_FILTER_ATTR_NAME
Specify the value for 'fcst.filter_attr_name' in the MET configuration file for MODE.

Used by: MODE

MODE_FCST_FILTER_ATTR_THRESH
Specify the value for 'fcst.filter_attr_thresh' in the MET configuration file for MODE.

Used by: MODE

MODE_FCST_MERGE_FLAG
Sets the merge_flag value in the mode config file for forecast fields. Valid values are NONE, THRESH, ENGINE, and BOTH.

Used by: MODE

MODE_FCST_MERGE_THRESH
Comma separated list of merge threshold values used by mode for forecast fields.

Used by: MODE

MODE_FCST_MULTIVAR_LEVEL
Specify the value for 'fcst.multivar_level' in the MET configuration file for MODE.

Used by: MODE

MODE_FCST_MULTIVAR_NAME
Specify the value for 'fcst.multivar_name' in the MET configuration file for MODE.

Used by: MODE

MODE_FCST_VLD_THRESH
Specify the value for 'fcst.vld_thresh' in the MET configuration file for MODE.

Used by: MODE

MODE_GRID_RES
Set the grid_res entry in the MODE MET config file.

Used by: MODE

MODE_INTEREST_FUNCTION_BOUNDARY_DIST
Specify the value for 'interest_function.boundary_dist' in the MET configuration file for MODE.

Used by: MODE

MODE_INTEREST_FUNCTION_CENTROID_DIST
Specify the value for 'interest_function.centroid_dist' in the MET configuration file for MODE.

Used by: MODE

MODE_INTEREST_FUNCTION_CONVEX_HULL_DIST
Specify the value for 'interest_function.convex_hull_dist' in the MET configuration file for MODE.

Used by: MODE

MODE_MASK_GRID
Specify the value for 'mask.grid' in the MET configuration file for MODE.

Used by: MODE

MODE_MASK_GRID_FLAG
Specify the value for 'mask.grid_flag' in the MET configuration file for MODE.

Used by: MODE

MODE_MASK_MISSING_FLAG
Specify the value for 'mask_missing_flag' in the MET configuration file for MODE.

Used by: MODE

MODE_MASK_POLY
Specify the value for 'mask.poly' in the MET configuration file for MODE.

Used by: MODE

MODE_MASK_POLY_FLAG
Specify the value for 'mask.poly_flag' in the MET configuration file for MODE.

Used by: MODE

MODE_MATCH_FLAG
Specify the value for 'match_flag' in the MET configuration file for MODE.

Used by: MODE

MODE_MAX_CENTROID_DIST
Specify the value for 'max_centroid_dist' in the MET configuration file for MODE.

Used by: MODE

MODE_MERGE_CONFIG_FILE
Path to mode merge config file.

Used by: MODE

MODE_MERGE_FLAG
Sets the merge_flag value in the mode config file for both forecast and observation fields. Has the same behavior as setting *MODE_FCST_MERGE_FLAG* and *MODE_OBS_MERGE_FLAG* to the same value. Valid values are NONE, THRESH, ENGINE, and BOTH.

Used by: MODE

MODE_MERGE_THRESH
Comma separated list of merge threshold values used by mode for forecast and observation fields. Has the same behavior as setting *MODE_FCST_MERGE_THRESH* and *MODE_OBS_MERGE_THRESH* to the same value.

Used by: MODE

MODE_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: MODE_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: MODE

MODE_MULTIVAR_INTENSITY_FLAG
Specify the value for 'multivar_intensity_flag' in the MET configuration file for MODE.

Used by: MODE

MODE_MULTIVAR_LOGIC
Specify the value for 'multivar_logic' in the MET configuration file for MODE. If this variable is set, then multi-variate MODE will be run. This means that more than 1 input file will be read and all of

the fields specified will be processed in a single call to MODE. See the MET User's Guide for more information on multi-variate MODE.

Used by: MODE

MODE_NC_PAIRS_FLAG_CLUSTER_ID
Specify the value for 'nc_pairs_flag.cluster_id' in the MET configuration file for MODE.

Used by: MODE

MODE_NC_PAIRS_FLAG_LATLON
Specify the value for 'nc_pairs_flag.latlon' in the MET configuration file for MODE.

Used by: MODE

MODE_NC_PAIRS_FLAG_OBJECT_ID
Specify the value for 'nc_pairs_flag.object_id' in the MET configuration file for MODE.

Used by: MODE

MODE_NC_PAIRS_FLAG_OBJECT_RAW
Specify the value for 'nc_pairs_flag.object_raw' in the MET configuration file for MODE.

Used by: MODE

MODE_NC_PAIRS_FLAG_POLYLINES
Specify the value for 'nc_pairs_flag.polylines' in the MET configuration file for MODE.

Used by: MODE

MODE_NC_PAIRS_FLAG_RAW
Specify the value for 'nc_pairs_flag.raw' in the MET configuration file for MODE.

Used by: MODE

MODE_OBS_CENSOR_THRESH
Specify the value for 'obs.censor_thresh' in the MET configuration file for MODE.

Used by: MODE

MODE_OBS_CENSOR_VAL
Specify the value for 'obs.censor_val' in the MET configuration file for MODE.

Used by: MODE

MODE_OBS_CONV_RADIUS

> **Warning:** **DEPRECATED:** Please see MET User's Guide instead.

MODE_OBS_CONV_THRESH

> **Warning:** **DEPRECATED:** Please use *OBS_MODE_CONV_THRESH* instead.

MODE_OBS_FILE_TYPE
Specify the value for 'obs.file_type' in the MET configuration file for MODE.

Used by: MODE

MODE_OBS_FILTER_ATTR_NAME
Specify the value for 'obs.filter_attr_name' in the MET configuration file for MODE.

Used by: MODE

MODE_OBS_FILTER_ATTR_THRESH
Specify the value for 'obs.filter_attr_thresh' in the MET configuration file for MODE.

Used by: MODE

MODE_OBS_MERGE_FLAG

> **Warning:** **DEPRECATED:** Please use *OBS_MODE_MERGE_FLAG* instead.

MODE_OBS_MERGE_THRESH

> **Warning:** **DEPRECATED:** Please use *OBS_MODE_MERGE_THRESH* instead.

MODE_OBS_MULTIVAR_LEVEL
Specify the value for 'obs.multivar_level' in the MET configuration file for MODE.

Used by: MODE

MODE_OBS_MULTIVAR_NAME
Specify the value for 'obs.multivar_name' in the MET configuration file for MODE.

Used by: MODE

MODE_OBS_VLD_THRESH
Specify the value for 'obs.vld_thresh' in the MET configuration file for MODE.

Used by: MODE

MODE_OUT_DIR

> **Warning:** **DEPRECATED:** Please use *MODE_OUTPUT_DIR* instead.

MODE_OUTPUT_DIR
Output directory to write mode files.

Used by: MODE

MODE_OUTPUT_PREFIX
String to pass to the MET config file to prepend text to the output filenames.

Used by: MODE

MODE_OUTPUT_TEMPLATE
Sets the subdirectories below *MODE_OUTPUT_DIR* using a template to allow run time information. If LOOP_BY = VALID, default value is valid time YYYYMMDDHHMM/mode. If LOOP_BY = INIT, default value is init time YYYYMMDDHHMM/mode.

Used by: MODE

MODE_PS_PLOT_FLAG
Specify the value for 'ps_plot_flag' in the MET configuration file for MODE.

Used by: MODE

MODE_QUILT
True/False. If True, run all permutations of radius and threshold.

Used by: MODE

MODE_REGRID_CENSOR_THRESH
Specify the value for 'regrid.censor_thresh' in the MET configuration file for MODE.

Used by: MODE

MODE_REGRID_CENSOR_VAL
Specify the value for 'regrid.censor_val' in the MET configuration file for MODE.

Used by: MODE

MODE_REGRID_CONVERT
Specify the value for 'regrid.convert' in the MET configuration file for MODE.

Used by: MODE

MODE_REGRID_METHOD
Specify the value for 'regrid.method' in the MET configuration file for MODE.

Used by: MODE

MODE_REGRID_SHAPE
Specify the value for 'regrid.shape' in the MET configuration file for MODE.

Used by: MODE

MODE_REGRID_TO_GRID
Used to set the regrid dictionary item 'to_grid' in the MET MODE config file. See the MET User's Guide for more information.

Used by: MODE

MODE_REGRID_VLD_THRESH
Specify the value for 'regrid.vld_thresh' in the MET configuration file for MODE.

Used by: MODE

MODE_REGRID_WIDTH
Specify the value for 'regrid.width' in the MET configuration file for MODE.

Used by: MODE

MODE_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: MODE

MODE_TOTAL_INTEREST_THRESH
Specify the value for 'total_interest_thresh' in the MET configuration file for MODE.

Used by: MODE

MODE_VERIFICATION_MASK_TEMPLATE
Template used to specify the verification mask filename for the MET tool mode. Now supports a list of filenames.

Used by: MODE

MODE_WEIGHT_ANGLE_DIFF
Specify the value for 'weight.angle_diff' in the MET configuration file for MODE.

Used by: MODE

MODE_WEIGHT_AREA_RATIO
Specify the value for 'weight.area_ratio' in the MET configuration file for MODE.

Used by: MODE

MODE_WEIGHT_ASPECT_DIFF
Specify the value for 'weight.aspect_diff' in the MET configuration file for MODE.

Used by: MODE

MODE_WEIGHT_BOUNDARY_DIST
Specify the value for 'weight.boundary_dist' in the MET configuration file for MODE.

Used by: MODE

MODE_WEIGHT_CENTROID_DIST
Specify the value for 'weight.centroid_dist' in the MET configuration file for MODE.

Used by: MODE

MODE_WEIGHT_COMPLEXITY_RATIO
Specify the value for 'weight.complexity_ratio' in the MET configuration file for MODE.

Used by: MODE

MODE_WEIGHT_CONVEX_HULL_DIST
Specify the value for 'weight.convex_hull_dist' in the MET configuration file for MODE.

Used by: MODE

MODE_WEIGHT_CURVATURE_RATIO
Specify the value for 'weight.curvature_ratio' in the MET configuration file for MODE.

Used by: MODE

MODE_WEIGHT_INT_AREA_RATIO
Specify the value for 'weight.int_area_ratio' in the MET configuration file for MODE.

Used by: MODE

MODE_WEIGHT_INTEN_PERC_RATIO
Specify the value for 'weight.inten_perc_ratio' in the MET configuration file for MODE.

Used by: MODE

MODE_WEIGHT_INTEN_PERC_VALUE
Specify the value for 'weight.inten_perc_value' in the MET configuration file for MODE.

Used by: MODE

MODEL
Specify the model name. This is the model name listed in the MET .stat files.

Used by: EnsembleStat, GridStat, PointStat, PCPCombine, TCPairs, GridDiag, TCRMW

MODEL<n>
Define the model name for the first model to be used in the analysis. This is the model name listed in the MET .stat files.There can be <n> number of models defined in configuration files, simply increment the "MODEL1" string to match the total number of models being used, e.g.:

MODEL1
MODEL2
...
MODEL<n>

Used by: StatAnalysis

MODEL<n>_NAME

> **Warning: DEPRECATED:** Please use *MODEL<n>*.

MODEL<n>_OBS_NAME

> **Warning: DEPRECATED:** Please use *MODEL<n>_OBTYPE* instead.

MODEL<n>_OBTYPE
Define the observation name that was used to compare the first model to be. This is the observation name listed in the MET .stat files. There can be <n> number of observation names defined in configuration files, simply increment the "MODEL1" string to match the total number of models being used, e.g.:

MODEL1_OBTYPE
MODEL2_OBTYPE
...
MODEL<n>_OBTYPE

Used by: StatAnalysis

MODEL<n>_REFERENCE_NAME

> **Warning: DEPRECATED:** No longer used.

MODEL<n>_STAT_ANALYSIS_DUMP_ROW_TEMPLATE
Specify the template to use for the stat_analysis dump_row file. A user customized template to use for the dump_row file. If left blank and a dump_row file is requested, a default version will be used.

Used by: StatAnalysis

MODEL<n>_STAT_ANALYSIS_LOOKIN_DIR

Specify the input directory where the MET stat_analysis tool will find input files. This is the directory that the stat_analysis wrapper will use to build the argument to -lookin for the MET stat_analysis tool. It can contain wildcards, i.e. *.

Used by: StatAnalysis

MODEL<n>_STAT_ANALYSIS_OUT_STAT_TEMPLATE

Specify the template to use for the stat_analysis out_stat file. A user customized template to use for the out_stat file. If left blank and a out_stat file is requested, a default version will be used.

Used by: StatAnalysis

MODEL<n>_STAT_DIR

Warning: **DEPRECATED:** Please use *MODEL<n>_STAT_ANALYSIS_LOOKIN_DIR* instead.

MODEL_DATA_DIR

Warning: **DEPRECATED:** Please use *EXTRACT_TILES_GRID_INPUT_DIR* instead.

MODEL_LIST

List of the specified the model names. If this is left unset, then values from *MODEL<n>* will be used.

Groups of values can be looped over by setting MODEL_LIST<n> and adding MODEL_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

MODEL_NAME

Warning: **DEPRECATED:** Please use *MODEL* instead.

MTD_CONFIG

Warning: **DEPRECATED:** Please use *MTD_CONFIG_FILE* instead.

MTD_CONFIG_FILE
Path to configuration file read by mtd. If unset, parm/met_config/MTDConfig_wrapped will be used.

Used by: MTD

MTD_CONV_RADIUS
Comma separated list of convolution radius values used by mode-TD for both forecast and observation files. Has the same behavior as setting *FCST_MTD_CONV_RADIUS* and *OBS_MTD_CONV_RADIUS* to the same value.

Used by: MTD

MTD_CONV_THRESH
Comma separated list of convolution threshold values used by mode-TD for both forecast and observation files. Has the same behavior as setting *FCST_MTD_CONV_THRESH* and *OBS_MTD_CONV_THRESH* to the same value.

Used by: MTD

MTD_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: MTD

MTD_DESC
Specify the value for 'desc' in the MET configuration file for MTD.

Used by: MTD

MTD_FCST_CONV_RADIUS
Comma separated list of convolution radius values used by mode-TD for forecast files.

Used by: MTD

MTD_FCST_CONV_THRESH
Comma separated list of convolution threshold values used by mode-TD for forecast files.

Used by: MTD

MTD_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: MTD_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: MTD

MTD_MIN_VOLUME
Sets min_volume in the MET MODE-TD config file. Refer to the MET User's Guide for more information.

Used by: MTD

MTD_OBS_CONV_RADIUS
Comma separated list of convolution radius values used by mode-TD for observation files.

Used by: MTD

MTD_OBS_CONV_THRESH
Comma separated list of convolution threshold values used by mode-TD for observation files.

Used by: MTD

MTD_OUT_DIR

Warning: DEPRECATED: Please use *MTD_OUTPUT_DIR*.

MTD_OUTPUT_DIR
Output directory to write mode-TD files.

Used by: MTD

MTD_OUTPUT_PREFIX
String to pass to the MET config file to prepend text to the output filenames.

Used by: MTD

MTD_OUTPUT_TEMPLATE
Sets the subdirectories below *MTD_OUTPUT_DIR* using a template to allow run time information. If LOOP_BY = VALID, default value is valid time YYYYMMDDHHMM/mtd. If LOOP_BY = INIT, default value is init time YYYYMMDDHHMM/mtd.

Used by: MTD

MTD_REGRID_CENSOR_THRESH
Specify the value for 'regrid.censor_thresh' in the MET configuration file for MTD.

Used by: MTD

MTD_REGRID_CENSOR_VAL
Specify the value for 'regrid.censor_val' in the MET configuration file for MTD.

Used by: MTD

MTD_REGRID_CONVERT
Specify the value for 'regrid.convert' in the MET configuration file for MTD.

Used by: MTD

MTD_REGRID_METHOD
Specify the value for 'regrid.method' in the MET configuration file for MTD.

Used by: MTD

MTD_REGRID_SHAPE
Specify the value for 'regrid.shape' in the MET configuration file for MTD.

Used by: MTD

MTD_REGRID_TO_GRID
Used to set the regrid dictionary item 'to_grid' in the MET MTD config file. See the MET User's Guide for more information.

Used by: MTD

MTD_REGRID_VLD_THRESH
Specify the value for 'regrid.vld_thresh' in the MET configuration file for MTD.

Used by: MTD

MTD_REGRID_WIDTH
Specify the value for 'regrid.width' in the MET configuration file for MTD.

Used by: MTD

MTD_SINGLE_DATA_SRC
Used only if MTD_SINGLE_RUN is set to True. Valid options are 'FCST' or 'OBS'.

Used by: MTD

MTD_SINGLE_RUN
Set to True to only process one data set (forecast or observation) in MODE-TD. If True, must set *MTD_SINGLE_RUN_SRC* to either 'FCST' or 'OBS'.

Used by: MTD

MTD_SINGLE_RUN_SRC

> **Warning:** **DEPRECATED:** Please use *MTD_SINGLE_DATA_SRC* instead.

MTD_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: MTD

NC_FILE_TMPL

> **Warning:** **DEPRECATED:** Please use *PB2NC_OUTPUT_TEMPLATE* instead.

NCDUMP
Path to thencdump executable.

Used by: PB2NC, PointStat

NCDUMP_EXE

> **Warning:** **DEPRECATED:** Please use *NCDUMP*.

NLAT

> **Warning:** **DEPRECATED:** Please use *EXTRACT_TILES_NLAT* instead.

NLON

> **Warning:** **DEPRECATED:** Please use *EXTRACT_TILES_NLON* instead.

NO_EE

> **Warning:** **DEPRECATED:** Please use *TCMPR_PLOTTER_NO_EE* instead.

NO_LOG

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_NO_LOG* instead.

OB_TYPE

Warning: **DEPRECATED:** Please use *OBTYPE* instead.

OBS_<n>_FIELD_NAME

Warning: **DEPRECATED:** Please use *OBS_PCP_COMBINE_<n>_FIELD_NAME* instead.

OBS_BUFR_VAR_LIST

Warning: **DEPRECATED:** Please use *PB2NC_OBS_BUFR_VAR_LIST* instead.

OBS_DATA_INTERVAL

Warning: **DEPRECATED:**

OBS_ENSEMBLE_STAT_FILE_WINDOW_BEGIN

Used to control the lower bound of the window around the valid time to determine if a file should be used for processing by EnsembleStat. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *OBS_ENSEMBLE_STAT_FILE_WINDOW_BEGIN* is not set in the config file, the value of *OBS_FILE_WINDOW_BEGIN* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_FILE_WINDOW_END

Used to control the upper bound of the window around the valid time to determine if a file should be used for processing by EnsembleStat. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *OBS_ENSEMBLE_STAT_FILE_WINDOW_END* is not set in the config file, the value of *OBS_FILE_WINDOW_END* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_GRID_INPUT_DIR
Input directory for grid observation files to use with the MET tool ensemble_stat. A similar variable exists for forecast data called *FCST_ENSEMBLE_STAT_INPUT_DIR*.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_GRID_INPUT_TEMPLATE
Template used to specify grid observation input filenames for the MET tool ensemble_stat. A similar variable exists for forecast data called *FCST_ENSEMBLE_STAT_INPUT_TEMPLATE*. To utilize Python Embedding as input to the MET tools, set this value to PYTHON_NUMPY or PYTHON_XARRAY.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_INPUT_GRID_DATATYPE
Specify the data type of the input directory for grid observation files used with the MET ensemble_stat tool. Currently valid options are NETCDF, GRIB, and GEMPAK. If set to GEMPAK, data will automatically be converted to NetCDF via GempakToCF. A similar variable exists for forecast data called *FCST_ENSEMBLE_STAT_INPUT_DATATYPE*.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_INPUT_POINT_DATATYPE
Specify the data type of the input directory for point observation files used with the MET ensemble_stat tool. Currently valid options are NETCDF, GRIB, and GEMPAK. If set to GEMPAK, data will automatically be converted to NetCDF via GempakToCF. A similar variable exists for forecast data called *FCST_ENSEMBLE_STAT_INPUT_DATATYPE*.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_POINT_INPUT_DIR
Input directory for point observation files to use with the MET tool ensemble_stat. A similar variable exists for forecast data called *FCST_ENSEMBLE_STAT_INPUT_DIR*.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_POINT_INPUT_TEMPLATE
Template used to specify point observation input filenames for the MET tool ensemble_stat. A similar

variable exists for forecast data called *FCST_ENSEMBLE_STAT_INPUT_TEMPLATE*. To utilize Python Embedding as input to the MET tools, set this value to PYTHON_PANDAS.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_VAR<n>_LEVELS
Wrapper specific field info variable. See *OBS_VAR<n>_LEVELS*.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_VAR<n>_NAME
Wrapper specific field info variable. See *OBS_VAR<n>_NAME*.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_VAR<n>_OPTIONS
Wrapper specific field info variable. See *OBS_VAR<n>_OPTIONS*.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_VAR<n>_THRESH
Wrapper specific field info variable. See *OBS_VAR<n>_THRESH*.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_WINDOW_BEGIN
Passed to the EnsembleStat MET config file to determine the range of data within a file that should be used for processing observation data. Units are seconds. If the variable is not set, EnsembleStat will use *OBS_WINDOW_BEGIN*.

Used by: EnsembleStat

OBS_ENSEMBLE_STAT_WINDOW_END
Passed to the EnsembleStat MET config file to determine the range of data within a file that should be used for processing observation data. Units are seconds. If the variable is not set, ensemble_stat will use *OBS_WINDOW_END*.

Used by: EnsembleStat

OBS_EXTRACT_TILES_INPUT_DIR
Directory containing gridded observation data to be used in ExtractTiles

Used by: ExtractTiles

OBS_EXTRACT_TILES_INPUT_TEMPLATE
Filename template used to identify observation input file to ExtractTiles.

Used by: ExtractTiles

OBS_EXTRACT_TILES_OUTPUT_TEMPLATE
Filename template used to identify the observation output file generated by ExtractTiles.

Used by: ExtractTiles

OBS_EXTRACT_TILES_PREFIX
Prefix for observation tile files. Used to create filename of intermediate files that are created while performing a series analysis.

Used by: ExtractTiles

OBS_FILE_WINDOW_BEGIN
Used to control the lower bound of the window around the valid time to determine if a file should be used for processing. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds.This value will be used for all wrappers that look for an observation file unless it is overridden by a wrapper specific configuration variable. For example, if *OBS_GRID_STAT_FILE_WINDOW_BEGIN* is set, the GridStat wrapper will use that value. If *PB2NC_FILE_WINDOW_BEGIN* is not set, then the PB2NC wrapper will use *OBS_FILE_WINDOW_BEGIN*.A corresponding variable exists for forecast data called *FCST_FILE_WINDOW_BEGIN*.

Used by: EnsembleStat, GridStat, MODE, MTD, PB2NC, PointStat

OBS_FILE_WINDOW_END
Used to control the upper bound of the window around the valid time to determine if a file should

be used for processing. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds.This value will be used for all wrappers that look for an observation file unless it is overridden by a wrapper specific configuration variable. For example, if *OBS_GRID_STAT_FILE_WINDOW_END* is set, the GridStat wrapper will use that value. If *PB2NC_FILE_WINDOW_END* is not set, then the PB2NC wrapper will use *OBS_FILE_WINDOW_END*. A corresponding variable exists for forecast data called *FCST_FILE_WINDOW_END*.

Used by: EnsembleStat, GridStat, MODE, MTD, PB2NC, PointStat

OBS_GEMPAK_INPUT_DIR

Warning: DEPRECATED: Please use *GEMPAKTOCF_INPUT_DIR* instead.

OBS_GEMPAK_TEMPLATE

Warning: DEPRECATED: Please use *GEMPAKTOCF_INPUT_TEMPLATE* instead.

OBS_GRID_STAT_FILE_TYPE
Specify the value for 'obs.file_type' in the MET configuration file for GridStat.

Used by: GridStat

OBS_GRID_STAT_FILE_WINDOW_BEGIN
Used to control the lower bound of the window around the valid time to determine if a file should be used for processing by GridStat. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *OBS_GRID_STAT_FILE_WINDOW_BEGIN* is not set in the config file, the value of *OBS_FILE_WINDOW_BEGIN* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by: GridStat

OBS_GRID_STAT_FILE_WINDOW_END
Used to control the upper bound of the window around the valid time to determine if a file should be used for processing by GridStat. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *OBS_GRID_STAT_FILE_WINDOW_END* is not set in the config file, the value of *OBS_FILE_WINDOW_END* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by: GridStat

OBS_GRID_STAT_INPUT_DATATYPE
See *FCST_GRID_STAT_INPUT_DATATYPE*

Used by: GridStat

OBS_GRID_STAT_INPUT_DIR
See *FCST_GRID_STAT_INPUT_DIR*

Used by: GridStat

OBS_GRID_STAT_INPUT_TEMPLATE
See *FCST_GRID_STAT_INPUT_TEMPLATE*

Used by: GridStat

OBS_GRID_STAT_PROB_THRESH
See *FCST_GRID_STAT_PROB_THRESH*

Used by: GridStat

OBS_GRID_STAT_VAR<n>_LEVELS
Wrapper specific field info variable. See *OBS_VAR<n>_LEVELS*.

Used by: GridStat

OBS_GRID_STAT_VAR<n>_NAME
Wrapper specific field info variable. See *OBS_VAR<n>_NAME*.

Used by: GridStat

OBS_GRID_STAT_VAR<n>_OPTIONS
Wrapper specific field info variable. See *OBS_VAR<n>_OPTIONS*.

Used by: GridStat

OBS_GRID_STAT_VAR<n>_THRESH
Wrapper specific field info variable. See *OBS_VAR<n>_THRESH*.

Used by: GridStat

OBS_GRID_STAT_WINDOW_BEGIN
Passed to the GridStat MET config file to determine the range of data within a file that should be used for processing. Units are seconds. If the variable is not set, GridStat will use *OBS_WINDOW_BEGIN*.

Used by: GridStat

OBS_GRID_STAT_WINDOW_END
Passed to the GridStat MET config file to determine the range of data within a file that should be used for processing. Units are seconds. If the variable is not set, GridStat will use *OBS_WINDOW_END*.

Used by: GridStat

OBS_INIT_HOUR_LIST
Specify a list of hours for initialization times of observation files for use in the analysis.

Groups of values can be looped over by setting OBS_INIT_HOUR_LIST<n> and adding OBS_INIT_HOUR_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

OBS_INPUT_DIR

Warning: **DEPRECATED:** Please use *OBS_POINT_STAT_INPUT_DIR* instead.

OBS_INPUT_DIR_REGEX

Warning: **DEPRECATED:** Please use *OBS_POINT_STAT_INPUT_DIR* instead.

OBS_INPUT_FILE_REGEX

> **Warning:** **DEPRECATED:** Please use *OBS_POINT_STAT_INPUT_TEMPLATE* instead.

OBS_INPUT_FILE_TMPL

> **Warning:** **DEPRECATED:** Please use *OBS_POINT_STAT_INPUT_TEMPLATE* instead.

OBS_IS_DAILY_FILE

> **Warning:** **DEPRECATED:**

OBS_IS_PROB

Specify whether the observation data are probabilistic or not. Used when setting OBS_* variables to process probabilistic forecast data. See *FCST_IS_PROB*

Used by: EnsembleStat, GridStat, MODE, MTD, PointStat, SeriesAnalysis

OBS_LEAD_LIST

Specify the values of the OBS_LEAD column in the MET .stat file to use. Comma separated list format, e.g.: 00, 24, 48, 72, 96, 120

Groups of values can be looped over by setting OBS_LEAD_LIST<n> and adding OBS_LEAD_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

OBS_LEVEL

> **Warning:** **DEPRECATED:** Please use *OBS_PCP_COMBINE_INPUT_LEVEL* instead.

OBS_LEVEL_LIST

Specify the values of the OBS_LEV column in the MET .stat file to use.

Groups of values can be looped over by setting OBS_LEVEL_LIST<n> and adding OBS_LEVEL_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

OBS_MAX_FORECAST

> **Warning:** **DEPRECATED:** Please use *OBS_PCP_COMBINE_MAX_FORECAST*.

OBS_MIN_FORECAST

> **Warning:** **DEPRECATED:** Please use *OBS_PCP_COMBINE_MIN_FORECAST*.

OBS_MODE_CONV_RADIUS
See *FCST_MODE_CONV_RADIUS*

Used by: MODE

OBS_MODE_CONV_THRESH
See *FCST_MODE_CONV_THRESH*

Used by: MODE

OBS_MODE_FILE_WINDOW_BEGIN
Used to control the lower bound of the window around the valid time to determine if a file should be used for processing by MODE. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *OBS_MODE_FILE_WINDOW_BEGIN* is not set in the config file, the value of *OBS_FILE_WINDOW_BEGIN* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by: MODE

OBS_MODE_FILE_WINDOW_END
Used to control the upper bound of the window around the valid time to determine if a file should be used for processing by MODE. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *OBS_MODE_FILE_WINDOW_END* is not set in the config file, the value of *OBS_FILE_WINDOW_END* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by: MODE

OBS_MODE_INPUT_DATATYPE
See *FCST_MODE_INPUT_DATATYPE*.

Used by: MODE

OBS_MODE_INPUT_DIR
See *FCST_MODE_INPUT_DIR*.

Used by: MODE

OBS_MODE_INPUT_TEMPLATE
See *FCST_MODE_INPUT_TEMPLATE*.

Used by: MODE

OBS_MODE_MERGE_FLAG
See *FCST_MODE_MERGE_FLAG*.

Used by: MODE

OBS_MODE_MERGE_THRESH
See *FCST_MODE_MERGE_THRESH*.

Used by: MODE

OBS_MODE_VAR<n>_LEVELS
Wrapper specific field info variable. See *OBS_VAR<n>_LEVELS*.

Used by: MODE

OBS_MODE_VAR<n>_NAME
Wrapper specific field info variable. See *OBS_VAR<n>_NAME*.

Used by: MODE

OBS_MODE_VAR<n>_OPTIONS
Wrapper specific field info variable. See *OBS_VAR<n>_OPTIONS*.

Used by: MODE

OBS_MODE_VAR<n>_THRESH
Wrapper specific field info variable. See *OBS_VAR<n>_THRESH*.

Used by: MODE

OBS_MODE_WINDOW_BEGIN
Passed to the MODE MET config file to determine the range of data within a file that should be used for processing. Units are seconds. If the variable is not set, MODE will use *OBS_WINDOW_BEGIN*.

Used by: MODE

OBS_MODE_WINDOW_END
Passed to the MODE MET config file to determine the range of data within a file that should be used for processing. Units are seconds. If the variable is not set, MODE will use *OBS_WINDOW_END*.

Used by: MODE

OBS_MTD_CONV_RADIUS
See *FCST_MTD_CONV_RADIUS*.

Used by: MTD

OBS_MTD_CONV_THRESH
See *FCST_MTD_CONV_THRESH*.

Used by: MTD

OBS_MTD_FILE_WINDOW_BEGIN
Used to control the lower bound of the window around the valid time to determine if a file should be used for processing by MTD. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *OBS_MTD_FILE_WINDOW_BEGIN* is not set

in the config file, the value of *OBS_FILE_WINDOW_BEGIN* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by:

OBS_MTD_FILE_WINDOW_END
Used to control the upper bound of the window around the valid time to determine if a file should be used for processing by MTD. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *OBS_MTD_FILE_WINDOW_END* is not set in the config file, the value of *OBS_FILE_WINDOW_END* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by: MTD

OBS_MTD_INPUT_DATATYPE
See *FCST_MTD_INPUT_DATATYPE*.

Used by: MTD

OBS_MTD_INPUT_DIR
See *FCST_MTD_INPUT_DIR*.

Used by: MTD

OBS_MTD_INPUT_FILE_LIST
Specifies an explicit path to a file list file to pass into mtd with the -obs or -single argument. If set, *OBS_MTD_INPUT_TEMPLATE* and *OBS_MTD_INPUT_DIR* are ignored. See also *FCST_MTD_INPUT_FILE_LIST*.

Used by: MTD

OBS_MTD_INPUT_TEMPLATE
See *FCST_MTD_INPUT_TEMPLATE*.

Used by:

OBS_MTD_VAR<n>_LEVELS
Wrapper specific field info variable. See *OBS_VAR<n>_LEVELS*.

Used by: MTD

OBS_MTD_VAR<n>_NAME
Wrapper specific field info variable. See *OBS_VAR<n>_NAME*.

Used by: MTD

OBS_MTD_VAR<n>_OPTIONS
Wrapper specific field info variable. See *OBS_VAR<n>_OPTIONS*.

Used by: MTD

OBS_MTD_VAR<n>_THRESH
Wrapper specific field info variable. See *OBS_VAR<n>_THRESH*.

Used by: MTD

OBS_NAME

> **Warning:** **DEPRECATED:** No longer used. Provide a string to identify the observation dataset name.

OBS_NATIVE_DATA_TYPE

> **Warning:** **DEPRECATED:** Please use *OBS_PCP_COMBINE_INPUT_DATATYPE* instead.

OBS_PCP_COMBINE_<n>_FIELD_NAME
See *FCST_PCP_COMBINE_<n>_FIELD_NAME*.

Used by: PCPCombine

OBS_PCP_COMBINE_BUCKET_INTERVAL
See *FCST_PCP_COMBINE_BUCKET_INTERVAL*.

Used by: PCPCombine

OBS_PCP_COMBINE_COMMAND
Used only when *OBS_PCP_COMBINE_METHOD* = USER_DEFINED. Custom command to run PCPCombine with a complex call that doesn't fit common use cases. Value can include filename template syntax, i.e. {valid?fmt=%Y%m%d}, that will be substituted based on the current runtime. The name of the application and verbosity flag does not need to be included. For example, if set to '-derive min,max /some/file' the command run will be pcp_combine -v 2 -derive min,max /some/file. A corresponding variable exists for forecast data called *FCST_PCP_COMBINE_COMMAND*.

Used by: PCPCombine

OBS_PCP_COMBINE_CONSTANT_INIT
If True, only look for observation files that have a given initialization time. Used only if *OBS_PCP_COMBINE_INPUT_TEMPLATE* has a 'lead' tag. If set to False, the lowest forecast lead for each search (valid) time is used. This variable is only used if model data is used as the OBS to compare to other model data as the FCST.

Used by: PCPCombine

OBS_PCP_COMBINE_DATA_INTERVAL

Warning: DEPRECATED:

OBS_PCP_COMBINE_DERIVE_LOOKBACK
See *FCST_PCP_COMBINE_DERIVE_LOOKBACK*.

Used by: PCPCombine

OBS_PCP_COMBINE_EXTRA_LEVELS
See *FCST_PCP_COMBINE_EXTRA_LEVELS*

Used by: PCPCombine

OBS_PCP_COMBINE_EXTRA_NAMES
See *FCST_PCP_COMBINE_EXTRA_NAMES*

Used by: PCPCombine

OBS_PCP_COMBINE_EXTRA_OUTPUT_NAMES
See *FCST_PCP_COMBINE_EXTRA_OUTPUT_NAMES*

Used by: PCPCombine

OBS_PCP_COMBINE_INPUT_ACCUMS
See *FCST_PCP_COMBINE_INPUT_ACCUMS*

Used by: PCPCombine

OBS_PCP_COMBINE_INPUT_DATATYPE
See *FCST_PCP_COMBINE_INPUT_DATATYPE*.

Used by: PCPCombine

OBS_PCP_COMBINE_INPUT_DIR
See *FCST_PCP_COMBINE_INPUT_DIR*.

Used by: PCPCombine

OBS_PCP_COMBINE_INPUT_LEVEL
See *FCST_PCP_COMBINE_INPUT_LEVEL*.

Used by: PCPCombine

OBS_PCP_COMBINE_INPUT_LEVELS
See *FCST_PCP_COMBINE_INPUT_LEVELS*

Used by: PCPCombine

OBS_PCP_COMBINE_INPUT_NAMES
See *FCST_PCP_COMBINE_INPUT_NAMES*

Used by: PCPCombine

OBS_PCP_COMBINE_INPUT_OPTIONS
See *FCST_PCP_COMBINE_INPUT_OPTIONS*

Used by: PCPCombine

OBS_PCP_COMBINE_INPUT_TEMPLATE
See *FCST_PCP_COMBINE_INPUT_TEMPLATE*.

Used by: PCPCombine

OBS_PCP_COMBINE_IS_DAILY_FILE

Warning: DEPRECATED:

OBS_PCP_COMBINE_LOOKBACK
See *FCST_PCP_COMBINE_LOOKBACK*.

Used by: PCPCombine

OBS_PCP_COMBINE_MAX_FORECAST
See *FCST_PCP_COMBINE_MAX_FORECAST*.

Used by: PCPCombine

OBS_PCP_COMBINE_METHOD
See *FCST_PCP_COMBINE_METHOD*.

Used by: PCPCombine

OBS_PCP_COMBINE_MIN_FORECAST
See *FCST_PCP_COMBINE_MIN_FORECAST*.

Used by: PCPCombine

OBS_PCP_COMBINE_OUTPUT_ACCUM
See *FCST_PCP_COMBINE_LOOKBACK*.

Used by: PCPCombine

OBS_PCP_COMBINE_OUTPUT_DIR
See *FCST_PCP_COMBINE_OUTPUT_DIR*.

Used by: PCPCombine

OBS_PCP_COMBINE_OUTPUT_NAME
See *FCST_PCP_COMBINE_OUTPUT_NAME*.

Used by: PCPCombine

OBS_PCP_COMBINE_OUTPUT_TEMPLATE
See *FCST_PCP_COMBINE_OUTPUT_TEMPLATE*.

Used by: PCPCombine

OBS_PCP_COMBINE_RUN
See *FCST_PCP_COMBINE_RUN*. Acceptable values: true/false

Used by: PCPCombine

OBS_PCP_COMBINE_STAT_LIST
See *FCST_PCP_COMBINE_STAT_LIST*. Acceptable values: sum, min, max, range, mean, stdev, vld_count

Used by: PCPCombine

OBS_PCP_COMBINE_TIMES_PER_FILE

> **Warning: DEPRECATED:**

OBS_PCP_COMBINE_USE_ZERO_ACCUM
Only used if running PCPCombine wrapper with *OBS_PCP_COMBINE_METHOD* = SUBTRACT. See *FCST_PCP_COMBINE_USE_ZERO_ACCUM* for more information.

Used by: PCPCombine

OBS_POINT_STAT_FILE_WINDOW_BEGIN
Used to control the lower bound of the window around the valid time to determine if a file should be used for processing by PointStat. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *OBS_POINT_STAT_FILE_WINDOW_BEGIN* is not set in the config file, the value of *OBS_FILE_WINDOW_BEGIN* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by: PointStat

OBS_POINT_STAT_FILE_WINDOW_END
Used to control the upper bound of the window around the valid time to determine if a file should be used for processing by PointStat. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *OBS_POINT_STAT_FILE_WINDOW_END* is not set in the config file, the value of *OBS_FILE_WINDOW_END* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by: PointStat

OBS_POINT_STAT_INPUT_DATATYPE
See *FCST_POINT_STAT_INPUT_DATATYPE*.

Used by: PointStat

OBS_POINT_STAT_INPUT_DIR
See *FCST_POINT_STAT_INPUT_DIR*.

Used by: PointStat

OBS_POINT_STAT_INPUT_TEMPLATE
See *FCST_POINT_STAT_INPUT_TEMPLATE*.

Used by: GriPointStat

OBS_POINT_STAT_VAR<n>_LEVELS
Wrapper specific field info variable. See *OBS_VAR<n>_LEVELS*.

Used by: PointStat

OBS_POINT_STAT_VAR<n>_NAME
Wrapper specific field info variable. See *OBS_VAR<n>_NAME*.

Used by: PointStat

OBS_POINT_STAT_VAR<n>_OPTIONS
Wrapper specific field info variable. See *OBS_VAR<n>_OPTIONS*.

Used by: PointStat

OBS_POINT_STAT_VAR<n>_THRESH
Wrapper specific field info variable. See *OBS_VAR<n>_THRESH*.

Used by: PointStat

OBS_POINT_STAT_WINDOW_BEGIN
Passed to the PointStat MET config file to determine the range of data within a file that should be used for processing observation data. Units are seconds. If the variable is not set, PointStat will use *OBS_WINDOW_BEGIN*.

Used by: PointStat

OBS_POINT_STAT_WINDOW_END
Passed to the PointStat MET config file to determine the range of data within a file that should be used for processing observation data. Units are seconds. If the variable is not set, PointStat will use *OBS_WINDOW_END*.

Used by: PointStat

OBS_PROB_IN_GRIB_PDS
Boolean to specify whether the probabilistic forecast data is stored in the GRIB Product Definition Section or not. Used when setting OBS_* variables to process probabilistic forecast data. Only used when *OBS_IS_PROB* is True. See *FCST_PROB_IN_GRIB_PDS* and *FCST_IS_PROB*.

Used by: EnsembleStat, GridStat, MODE, MTD, PointStat, SeriesAnalysis

OBS_REGRID_DATA_PLANE_INPUT_DATATYPE
See *FCST_REGRID_DATA_PLANE_INPUT_DATATYPE*.

Used by: RegridDataPlane

OBS_REGRID_DATA_PLANE_INPUT_DIR
See *FCST_REGRID_DATA_PLANE_INPUT_DIR*.

Used by: RegridDataPlane

OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE
See *FCST_REGRID_DATA_PLANE_INPUT_TEMPLATE*.

Used by: RegridDataPlane

OBS_REGRID_DATA_PLANE_OUTPUT_DIR
See *FCST_REGRID_DATA_PLANE_OUTPUT_DIR*.

Used by: RegridDataPlane

OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE
See *FCST_REGRID_DATA_PLANE_OUTPUT_TEMPLATE*.

Used by: RegridDataPlane

OBS_REGRID_DATA_PLANE_RUN
If True, process observation data with RegridDataPlane.

Used by: RegridDataPlane

OBS_REGRID_DATA_PLANE_TEMPLATE
See *FCST_REGRID_DATA_PLANE_TEMPLATE*.

Used by: RegridDataPlane

OBS_REGRID_DATA_PLANE_VAR<n>_INPUT_FIELD_NAME
Specify the (optional) observation input field name that is created by RegridDataPlane. The name corresponds to *OBS_VAR<n>_NAME*. This is used when using Python Embedding as input to the MET tool, because the *OBS_VAR<n>_NAME* defines the python script to call.

Used by: RegridDataPlane

OBS_REGRID_DATA_PLANE_VAR<n>_INPUT_LEVEL
Specify the (optional) observation input field level that is created by RegridDataPlane. The name corresponds to *OBS_VAR<n>_LEVELS*. This is used when using Python Embedding as input to the MET tool, because the *OBS_VAR<n>_LEVELS* defines the python script to call.

Used by: RegridDataPlane

OBS_REGRID_DATA_PLANE_VAR<n>_OUTPUT_FIELD_NAME
Specify the observation output field name that is created by RegridDataPlane. The name corresponds to *OBS_VAR<n>_NAME*. This is used when using Python Embedding as input to the MET tool, because the *OBS_VAR<n>_NAME* defines the python script to call.

Used by: RegridDataPlane

OBS_SERIES_ANALYSIS_ASCII_REGEX_LEAD

Warning: **DEPRECATED:** Please use *OBS_EXTRACT_TILES_PREFIX* instead.

OBS_SERIES_ANALYSIS_CAT_THRESH
Specify the value for 'obs.cat_thresh' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

OBS_SERIES_ANALYSIS_INPUT_DATATYPE
Set the file_type entry of the obs dictionary in the MET config file for SeriesAnalysis.

Used by: SeriesAnalysis

OBS_SERIES_ANALYSIS_INPUT_DIR
Specify the directory to read observation input in SeriesAnalysis. See also *OBS_SERIES_ANALYSIS_INPUT_TEMPLATE*

Used by: SeriesAnalysis

OBS_SERIES_ANALYSIS_INPUT_FILE_LIST
Specifies an explicit path to a file list file to pass into series_analysis with the -obs argument. If set, *FCST_SERIES_ANALYSIS_INPUT_FILE_LIST* must also be set and *OBS_SERIES_ANALYSIS_INPUT_TEMPLATE* and *OBS_SERIES_ANALYSIS_INPUT_DIR* are ignored. See also *BOTH_SERIES_ANALYSIS_INPUT_FILE_LIST*.

Used by: SeriesAnalysis

OBS_SERIES_ANALYSIS_INPUT_TEMPLATE
Template to find observation input in SeriesAnalysis. See also *OBS_SERIES_ANALYSIS_INPUT_DIR*

Used by: SeriesAnalysis

OBS_SERIES_ANALYSIS_NC_TILE_REGEX

Warning: **DEPRECATED:** Please use *OBS_EXTRACT_TILES_PREFIX* instead.

OBS_SERIES_ANALYSIS_TILE_INPUT_DIR

> **Warning: DEPRECATED:** Please use *OBS_SERIES_ANALYSIS_INPUT_DIR* instead.

OBS_THRESH_LIST

Specify the values of the OBS_THRESH column in the MET .stat file to use.

Groups of values can be looped over by setting OBS_THRESH_LIST<n> and adding OBS_THRESH_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

OBS_TIMES_PER_FILE

> **Warning: DEPRECATED:**

OBS_UNITS_LIST

Specify the values of the OBS_UNITS column in the MET .stat file to use.

Groups of values can be looped over by setting OBS_UNITS_LIST<n> and adding OBS_UNITS_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

OBS_VALID_HOUR_LIST

Specify a list of hours for valid times of observation files for use in the analysis.

Groups of values can be looped over by setting OBS_VALID_HOUR_LIST<n> and adding OBS_VALID_HOUR_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

OBS_VAR

> **Warning: DEPRECATED:** Specify the string for the observation variable used in the analysis. See *OBS_VAR<n>_NAME*, *OBS_VAR<n>_LEVELS*, *OBS_VAR<n>_OPTIONS* and *OBS_VAR<n>_THRESH* where n = integer >= 1.

OBS_VAR<n>_LEVELS

Define the levels for the <n>th observation variable to be used in the analysis where <n> is an integer >= 1. The value can be a single item or a comma separated list of items. You can define NetCDF levels, such as (0,*,*), but you will need to surround these values with quotation marks so that the commas in the item are not interpreted as an item delimeter. Some examples:

OBS_VAR1_LEVELS = A06, P500
OBS_VAR2_LEVELS = "(0,*,*)", "(1,*,*)"

There can be <n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.:

OBS_VAR1_LEVELS
OBS_VAR2_LEVELS
...
OBS_VAR<n>_LEVELS

If *OBS_VAR<n>_LEVELS* is set, then *FCST_VAR<n>_LEVELS* must be set as well. If the same value applies to both forecast and observation data, use *BOTH_VAR<n>_LEVELS*.

See *Field Info* (page 58) for more information.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

OBS_VAR<n>_NAME

Define the name for the <n>th observation variable to be used in the analysis where <n> is an integer >= 1. If *OBS_VAR<n>_NAME* is set, then *FCST_VAR<n>_NAME* must be set. If the same value applies to both forecast and observation data, use *BOTH_VAR<n>_NAME*. There can be <n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.:

OBS_VAR1_NAME
OBS_VAR2_NAME
...
OBS_VAR<n>_NAME

This value can be set to a call to a python script with arguments to supply data to the MET tools via Python Embedding. Filename template syntax can be used here to specify time information of an input

file, i.e. {valid?fmt=%Y%m%d%H}. See the MET User's Guide for more information about Python Embedding in the MET tools.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

OBS_VAR<n>_OPTIONS

Define the options for the <n>th observation variable to be used in the analysis where <n> is an integer >= 1. These addition options will be applied to every name/level/threshold combination for VAR<n>. If OBS_VAR<n>_OPTIONS is not set but *FCST_VAR<n>_OPTIONS* is, the same information will be used for both variables. There can be <n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.:

OBS_VAR1_OPTIONS
OBS_VAR2_OPTIONS
...
OBS_VAR<n>_OPTIONS

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

OBS_VAR<n>_THRESH

Define the threshold(s) for the <n>th observation variable to be used in the analysis where <n> is an integer >= 1. The value can be a single item or a comma separated list of items that must start with a comparison operator (>,>=,==,!=,<,<=,gt,ge,eq,ne,lt,le). If *OBS_VAR<n>_THRESH* is not set but *FCST_VAR<n>_THRESH* is, the same information will be used for both variables. There can be <n> number of these variables defined in configuration files, simply increment the VAR1 string to match the total number of variables being used, e.g.:

OBS_VAR1_THRESH
OBS_VAR2_THRESH
...
OBS_VAR<n>_THRESH

If OBS_VAR<n>_THRESH is set, then *FCST_VAR<n>_THRESH* must be set as well. If the same value applies to both forecast and observation data, use *BOTH_VAR<n>_THRESH*.

See *Field Info* (page 58) for more information.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

OBS_VAR_LEVEL

> **Warning:** **DEPRECATED:** Please use *OBS_LEVEL_LIST* instead.

OBS_VAR_LIST

Specify the values of the OBS_VAR column in the MET .stat file to use.

Groups of values can be looped over by setting OBS_VAR_LIST<n> and adding OBS_VAR_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

OBS_VAR_NAME

> **Warning:** **DEPRECATED:** Please use *OBS_VAR_LIST* instead.

OBS_WINDOW_BEG

> **Warning:** **DEPRECATED:** Please use *OBS_WINDOW_BEGIN*.

OBS_WINDOW_BEGIN

Passed to the MET config file to determine the range of data within a file that should be used for processing.Units are seconds. This value will be used for all wrappers that look for an observation file unless it is overridden by a wrapper specific configuration variable. For example, if *OBS_POINT_STAT_WINDOW_BEGIN* is set, the PointStat wrapper will use that value. If *PB2NC_OBS_WINDOW_BEGIN* is not set, then the PB2NC wrapper will use *OBS_WINDOW_BEGIN*. A corresponding variable exists for forecast data called *FCST_WINDOW_BEGIN*.

Used by: PB2NC, PointStat

OBS_WINDOW_END

Passed to the MET config file to determine the range of data within a file that should be used for processing.Units are seconds. This value will be used for all wrappers that look for an observation file unless it is overridden by a wrapper specific configuration variable. For example, if *OBS_POINT_STAT_WINDOW_END* is set, the PointStat wrapper will use that value. If *PB2NC_OBS_WINDOW_END* is not set, then the PB2NC wrapper will use *OBS_WINDOW_END*. A corresponding variable exists for forecast data called *FCST_WINDOW_END*.

Used by: PB2NC, PointStat

OBTYPE

Provide a string to represent the type of observation data used in the analysis. This is the observation time listed in the MET .stat files and is used in setting output filename.

Used by: EnsembleStat, GridStat, MODE, MTD, PointStat

OMP_NUM_THREADS

Sets environment variable of the same name that determines the number of threads to use in the MET executables. Defaults to 1 thread. If the environment variable of the same name is already set in the user's environment, then that value will be used instead of the value set in the METplus configuration. A warning will be output if this is the case and the values differ between them.

Used by: All

OUTPUT_BASE

Provide a path to the top level output directory for METplus.

Used by: All

OVERWRITE_NC_OUTPUT

Warning: **DEPRECATED:** Please use *PB2NC_SKIP_IF_OUTPUT_EXISTS* instead.

OVERWRITE_TRACK

Warning: **DEPRECATED:** Please use *EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS* instead.

PARM_BASE

This variable will automatically be set by METplus when it is started. Specifies the top level METplus parameter file directory. You can override this value by setting the environment variable METPLUS_PARM_BASE to another directory containing a copy of the METPlus parameter file directory. If the environment variable is not set, the parm directory corresponding to the calling script is used. It is recommended that this variable is not set by the user. If it is set and is not equivalent to the value determined by METplus, execution will fail.

Used by: All

PB2NC_CONFIG_FILE
Path to configuration file read by pb2nc. If unset, parm/met_config/PB2NCConfig_wrapped will be used.

Used by: PB2NC

PB2NC_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: PB2NC

PB2NC_FILE_WINDOW_BEGIN
Used to control the lower bound of the window around the valid time to determine if a file should be used for processing by PB2NC. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *PB2NC_FILE_WINDOW_BEGIN* is not set in the config file, the value of *OBS_FILE_WINDOW_BEGIN* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by: PB2NC

PB2NC_FILE_WINDOW_END
Used to control the upper bound of the window around the valid time to determine if a file should be used for processing by PB2NC. See *Directory and Filename Template Info* (page 65) subsection called 'Using Windows to Find Valid Files.' Units are seconds. If *PB2NC_FILE_WINDOW_END* is not set in the config file, the value of *OBS_FILE_WINDOW_END* will be used instead. If both file window begin and window end values are set to 0, then METplus will require an input file with an exact time match to process.

Used by: PB2NC

PB2NC_GRID
Specify a grid to use with the MET pb2nc tool.

Used by: PB2NC

PB2NC_INPUT_DATATYPE
Specify the data type of the input directory for prepbufr files used with the MET pb2nc tool. Currently valid options are NETCDF, GRIB, and GEMPAK. If set to GEMPAK, data will automatically be converted to NetCDF via GempakToCF.

Used by: PB2NC

PB2NC_INPUT_DIR
Specify the input directory where the MET PB2NC tool will look for files.

Used by: PB2NC

PB2NC_INPUT_TEMPLATE
Filename template of the input file used by PB2NC. See also *PB2NC_INPUT_DIR*.

Used by: PB2NC

PB2NC_LEVEL_CATEGORY
Specify the value for 'level_category' in the MET configuration file for PB2NC.

Used by: PB2NC

PB2NC_LEVEL_RANGE_BEG
Specify the value for 'level_range.beg' in the MET configuration file for PB2NC.

Used by: PB2NC

PB2NC_LEVEL_RANGE_END
Specify the value for 'level_range.end' in the MET configuration file for PB2NC.

Used by: PB2NC

PB2NC_MASK_GRID
Set the mask.grid entry in the PB2NC MET config file.

Used by: PN2NC

PB2NC_MASK_POLY
Set the mask.poly entry in the PB2NC MET config file.

Used by: PN2NC

PB2NC_MESSAGE_TYPE
Specify which PREPBUFR (PB) message types to convert using the MET pb2nc tool.

Used by: PB2NC

PB2NC_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: PB2NC_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: PB2NC

PB2NC_OBS_BUFR_MAP
Specify the value for 'obs_bufr_map' in the MET configuration file for PB2NC.

Used by: PB2NC

PB2NC_OBS_BUFR_VAR_LIST
Specify which BUFR codes to use from the observation dataset when using the MET pb2nc tool. Format is comma separated list, e.g.:PMO, TOB, TDO

Used by: PB2NC

PB2NC_OBS_WINDOW_BEGIN
Passed to the pb2nc MET config file to determine the range of data within a file that should be used for processing.Units are seconds. If the variable is not set, pb2nc will use *OBS_WINDOW_BEGIN*.

Used by: PB2NC

PB2NC_OBS_WINDOW_END
Passed to the pb2nc MET config file to determine the range of data within a file that should be used for processing. Units are seconds. If the variable is not set, pb2nc will use *OBS_WINDOW_END*.

Used by: PB2NC

PB2NC_OFFSETS
A list of potential offsets (in hours) that can be found in the *PB2NC_INPUT_TEMPLATE*. METplus will check if a file with a given offset exists in the order specified in this list, to be sure to put favored offset values first.

Used by: PB2NC

PB2NC_OUTPUT_DIR
Specify the directory where files will be written from the MET pb2nc tool.

Used by: PB2NC

PB2NC_OUTPUT_TEMPLATE
File template used to create netCDF files generated by PB2NC.

Used by: PB2NC

PB2NC_PB_REPORT_TYPE
Specify the value for 'pb_report_type' in the MET configuration file for PB2NC.

Used by: PB2NC

PB2NC_POLY

Note: please use *PB2NC_MASK_POLY*

Used by: PB2NC

PB2NC_QUALITY_MARK_THRESH
Specify the value for 'quality_mark_thresh' in the MET configuration file for PB2NC.

Used by: PB2NC

PB2NC_SKIP_IF_OUTPUT_EXISTS
If True, do not run PB2NC if output file already exists. Set to False to overwrite files.

Used by: PB2NC

PB2NC_STATION_ID
Specify the ID of the station to use with the MET PB2NC tool.

Used by: PB2NC

PB2NC_TIME_SUMMARY_BEG
Specify the time summary beg item in the MET pb2nc config file. Refer to the MET User's Guide for more information.

Used by: PB2NC

PB2NC_TIME_SUMMARY_END
Specify the time summary end item in the MET pb2nc config file. Refer to the MET User's Guide for more information.

Used by: PB2NC

PB2NC_TIME_SUMMARY_FLAG
Specify the time summary flag item in the MET pb2nc config file. Refer to the MET User's Guide for more information.

Used by: PB2NC

PB2NC_TIME_SUMMARY_GRIB_CODES
Specify the time summary grib_code item in the MET pb2nc config file. Refer to the MET User's Guide for more information.

Used by: PB2NC

PB2NC_TIME_SUMMARY_RAW_DATA
Specify the time summary raw_data item in the MET pb2nc config file. Refer to the MET User's Guide for more information.

Used by: PB2NC

PB2NC_TIME_SUMMARY_STEP
Specify the time summary step item in the MET pb2nc config file. Refer to the MET User's Guide for more information.

Used by: PB2NC

PB2NC_TIME_SUMMARY_TYPES
Specify the time summary type list item in the MET pb2nc config file. Refer to the MET User's Guide for more information.

Used by: PB2NC

PB2NC_TIME_SUMMARY_VALID_FREQ
Specify the time summary valid_freq item in the MET pb2nc config file. Refer to the MET User's Guide for more information.

Used by: PB2NC

PB2NC_TIME_SUMMARY_VALID_THRESH
Specify the time summary valid_thresh item in the MET pb2nc config file. Refer to the MET User's Guide for more information.

Used by: PN2NC

PB2NC_TIME_SUMMARY_VAR_NAMES
Specify the time summary obs_var list item in the MET pb2nc config file. Refer to the MET User's Guide for more information.

Used by: PB2NC

PB2NC_TIME_SUMMARY_WIDTH
Specify the time summary width item in the MET pb2nc config file. Refer to the MET User's Guide for more information.

Used by: PB2NC

PB2NC_VALID_BEGIN
Used to set the command line argument -valid_beg that controls the lower bound of valid times of data to use. Filename template notation can be used, i.e. {valid?fmt=%Y%m%d_%H%M%S}

Used by: PB2NC

PB2NC_VALID_END
Used to set the command line argument -valid_end that controls the upper bound of valid times of data to use. Filename template notation can be used, i.e. {valid?fmt=%Y%m%d_%H%M%S?shift=1d} (valid time shifted forward one day)

Used by: PB2NC

PB2NC_VERTICAL_LEVEL

> **Warning:** **DEPRECATED:** No longer used.

PCP_COMBINE_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: PCPCombine

PCP_COMBINE_METHOD

> **Warning:** **DEPRECATED:** Please use *OBS_PCP_COMBINE_METHOD* and/or *FCST_PCP_COMBINE_METHOD* instead.

PCP_COMBINE_SKIP_IF_OUTPUT_EXISTS
If True, do not run pcp_combine if output file already exists. Set to False to overwrite files.

Used by: PCPCombine

PLOT_CONFIG_OPTS

> **Warning: DEPRECATED:** Please use *TCMPR_PLOTTER_PLOT_CONFIG_OPTS* instead.

PLOT_DATA_PLANE_COLOR_TABLE
(Optional) path to color table file to override the default.

Used by: PlotDataPlane

PLOT_DATA_PLANE_CONVERT_TO_IMAGE
If set to True, run convert to create a png image with the same name as the output from plot_data_plane (except the extension is png instead of ps). If set to True, the application convert must either be in the user's path or [exe] CONVERT must be set to the full path to the executable.

Used by: PlotDataPlane

PLOT_DATA_PLANE_FIELD_EXTRA
Additional options for input field. Multiple options can be specified. Each option must end with a semi-colon including the last (or only) item.

Used by: PlotDataPlane

PLOT_DATA_PLANE_FIELD_LEVEL
Level of field to read from input file. For Python embedding input, do not set this value.

Used by: PlotDataPlane

PLOT_DATA_PLANE_FIELD_NAME
Name of field to read from input file. For Python embedding input, set to the path of a Python script and any arguments to the script.

Used by: PlotDataPlane

PLOT_DATA_PLANE_INPUT_DIR
Directory containing input data to PlotDataPlane. This variable is optional because you can specify the full path to the input files using *PLOT_DATA_PLANE_INPUT_TEMPLATE*.

Used by: PlotDataPlane

PLOT_DATA_PLANE_INPUT_TEMPLATE
Filename template of the input file used by PlotDataPlane. Set to PYTHON_NUMPY/XARRAY to read from a Python embedding script. See also *PLOT_DATA_PLANE_INPUT_DIR*.

Used by: PlotDataPlane

PLOT_DATA_PLANE_OUTPUT_DIR
Directory to write output data from PlotDataPlane. This variable is optional because you can specify the full path to the input files using *PLOT_DATA_PLANE_OUTPUT_TEMPLATE*.

Used by: PlotDataPlane

PLOT_DATA_PLANE_OUTPUT_TEMPLATE
Filename template of the output file created by PlotDataPlane. See also *PLOT_DATA_PLANE_OUTPUT_DIR*.

Used by: PlotDataPlane

PLOT_DATA_PLANE_RANGE_MIN_MAX
(Optional) minimum and maximum values to output to postscript file.

Used by: PlotDataPlane

PLOT_DATA_PLANE_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: PlotDataPlane

PLOT_DATA_PLANE_TITLE
(Optional) title to display on the output postscript file.

Used by: PlotDataPlane

PLOT_POINT_OBS_CENSOR_THRESH
Specify the value for 'censor_thresh' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_CENSOR_VAL
Specify the value for 'censor_val' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_DOTSIZE
Specify the value for 'dotsize' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_ELV_THRESH
Specify the value for 'elv_thresh' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_FILL_COLOR
Specify the value for 'fill_color' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_FILL_PLOT_INFO_COLOR_TABLE
Specify the value for 'fill_plot_info.color_table' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_FILL_PLOT_INFO_COLORBAR_FLAG
Specify the value for 'fill_plot_info.colorbar_flag' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_FILL_PLOT_INFO_FLAG
Specify the value for 'fill_plot_info.flag' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_FILL_PLOT_INFO_PLOT_MAX
Specify the value for 'fill_plot_info.plot_max' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_FILL_PLOT_INFO_PLOT_MIN
Specify the value for 'fill_plot_info.plot_min' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_DATA_FIELD
Specify the value for 'grid_data.field' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_COLOR_TABLE
Specify the value for 'grid_data.grid_plot_info.color_table' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_COLORBAR_FLAG
Specify the value for 'grid_data.grid_plot_info.colorbar_flag' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_PLOT_MAX
Specify the value for 'grid_data.grid_plot_info.plot_max' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_DATA_GRID_PLOT_INFO_PLOT_MIN
Specify the value for 'grid_data.grid_plot_info.plot_min' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_DATA_REGRID_METHOD
Specify the value for 'grid_data.regrid.method' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_DATA_REGRID_SHAPE
Specify the value for 'grid_data.regrid.shape' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_DATA_REGRID_TO_GRID
Specify the value for 'grid_data.regrid.to_grid' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_DATA_REGRID_VLD_THRESH
Specify the value for 'grid_data.regrid.vld_thresh' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_DATA_REGRID_WIDTH
Specify the value for 'grid_data.regrid.width' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_INPUT_DIR
Directory containing input grid data for PlotPointObs. This variable is optional because you can specify the full path to the input file using *PLOT_POINT_OBS_GRID_INPUT_TEMPLATE*.

Used by: PlotPointObs

PLOT_POINT_OBS_GRID_INPUT_TEMPLATE
Filename template of the input grid file for PlotPointObs. See also *PLOT_POINT_OBS_GRID_INPUT_DIR*.

Used by: PlotPointObs

PLOT_POINT_OBS_HGT_THRESH
Specify the value for 'hgt_thresh' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_INPUT_DIR
Directory containing input point observation data for PlotPointObs. This variable is optional because you can specify the full path to the input file(s) using *PLOT_POINT_OBS_INPUT_TEMPLATE*.

Used by: PlotPointObs

PLOT_POINT_OBS_INPUT_TEMPLATE
Filename template of the input point observation file(s) for PlotPointObs. See also *PLOT_POINT_OBS_INPUT_DIR*.

Used by: PlotPointObs

PLOT_POINT_OBS_LAT_THRESH
Specify the value for 'lat_thresh' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_LINE_COLOR
Specify the value for 'line_color' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_LINE_WIDTH
Specify the value for 'line_width' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_LON_THRESH
Specify the value for 'lon_thresh' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: PLOT_POINT_OBS_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: PlotPointObs

PLOT_POINT_OBS_MSG_TYP
Specify the value for 'msg_typ' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_OBS_GC
Specify the value for 'obs_gc' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_OBS_QUALITY
Specify the value for 'obs_quality' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_OBS_THRESH
Specify the value for 'obs_thresh' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_OBS_VAR
Specify the value for 'obs_var' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_OUTPUT_DIR
Directory containing output generated by PlotPointObs. This variable is optional because you can specify the full path to the output file using *PLOT_POINT_OBS_OUTPUT_TEMPLATE*.

Used by: PlotPointObs

PLOT_POINT_OBS_OUTPUT_TEMPLATE
Filename template of the output generated by PlotPointObs. See also *PLOT_POINT_OBS_OUTPUT_DIR*.

Used by: PlotPointObs

PLOT_POINT_OBS_POINT_DATA
Specify the value for 'point_data' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_PRS_THRESH
Specify the value for 'prs_thresh' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_RUNTIME_FREQ
Frequency to run PlotPointObs. See *Runtime Frequency* (page 70) for more information.

Used by: PlotPointObs

PLOT_POINT_OBS_SID_EXC
Specify the value for 'sid_exc' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_SID_INC
Specify the value for 'sid_inc' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_SKIP_IF_OUTPUT_EXISTS
If True, do not run plot_point_obs if output file already exists. Set to False to overwrite files.

Used by: PlotPointObs

PLOT_POINT_OBS_TITLE
Sets the title for the output plot generated by PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_VALID_BEG
Specify the value for 'valid_beg' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_POINT_OBS_VALID_END
Specify the value for 'valid_end' in the MET configuration file for PlotPointObs.

Used by: PlotPointObs

PLOT_TIME

Warning: **DEPRECATED:** Please use *DATE_TYPE* instead.

PLOT_TYPES

> **Warning:** **DEPRECATED:** Please use *TCMPR_PLOTTER_PLOT_TYPES* instead.

POINT2GRID_ADP
Provides an additional Aerosol Detection Product when GOES 16/17 input and an AOD variable name is used.

Used by: Point2Grid

POINT2GRID_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: Point2Grid

POINT2GRID_GAUSSIAN_DX
Gaussian dx value to add to the Point2Grid command line call with -gaussian_dx. Not added to call if unset or set to empty string.

Used by: Point2Grid

POINT2GRID_GAUSSIAN_RADIUS
Gaussian radius value to add to the Point2Grid command line call with -gaussian_radius. Not added to call if unset or set to empty string.

Used by: Point2Grid

POINT2GRID_INPUT_DIR
Directory containing the file containing point data used by point2grid. This variable is optional because you can specify the full path to a point file using *POINT2GRID_INPUT_TEMPLATE*.

Used by: Point2Grid

POINT2GRID_INPUT_FIELD
Specify the input field name that is read by Point2Grid.

Used by: Point2Grid

POINT2GRID_INPUT_LEVEL
Specify the input level name that is read by Point2Grid.

Used by: Point2Grid

POINT2GRID_INPUT_TEMPLATE
Filename template for the point file used by Point2Grid.

Used by: Point2Grid

POINT2GRID_OUTPUT_DIR
Specify the directory where output files from the MET point2grid tool are written.

Used by: Point2Grid

POINT2GRID_OUTPUT_TEMPLATE
Filename template for the output of Point2Grid.

Used by: Point2Grid

POINT2GRID_PROB_CAT_THRESH
Specify the probability threshold for practically perfect forecasts

Used by: Point2Grid

POINT2GRID_QC_FLAGS
Specify the qc flags name that is read by Point2Grid.

Used by: Point2Grid

POINT2GRID_REGRID_METHOD
Sets the gridding method used by point2grid.

Used by: Point2Grid

POINT2GRID_REGRID_TO_GRID
Used to set the grid definition for point2grid.

Used by: Point2Grid

POINT2GRID_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: Point2Grid

POINT2GRID_VLD_THRESH
Specify the required ratio of valid data for regridding

Used by: Point2Grid

POINT2GRID_WINDOW_BEGIN
Specify the beginning of the time window to use for a date stamp window to grab observations

Used by: Point2Grid

POINT2GRID_WINDOW_END
Specify the end of the time window to use for a date stamp window to grab observations

Used by: Point2Grid

POINT_STAT_CLIMO_CDF_BINS
Specify the value for 'climo_cdf.cdf_bins' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_CDF_CDF_BINS
See *POINT_STAT_CLIMO_CDF_BINS*

POINT_STAT_CLIMO_CDF_CENTER_BINS
Specify the value for 'climo_cdf.center_bins' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_CDF_DIRECT_PROB
Specify the value for 'climo_cdf.direct_prob' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_CDF_WRITE_BINS
Specify the value for 'climo_cdf.write_bins' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_DAY_INTERVAL
Specify the value for 'climo_mean.day_interval' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_FIELD
See: *<TOOL-NAME>_CLIMO_MEAN_FIELD*

Used by: PointStat

POINT_STAT_CLIMO_MEAN_FILE_NAME
Specify the value for 'climo_mean.file_name' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_HOUR_INTERVAL
Specify the value for 'climo_mean.hour_interval' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *POINT_STAT_CLIMO_MEAN_FILE_NAME*.

POINT_STAT_CLIMO_MEAN_INPUT_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *POINT_STAT_CLIMO_MEAN_FILE_NAME*.

POINT_STAT_CLIMO_MEAN_MATCH_MONTH
Specify the value for 'climo_mean.match_month' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_REGRID_METHOD
Specify the value for 'climo_mean.regrid.method' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_REGRID_SHAPE
Specify the value for 'climo_mean.regrid.shape' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_REGRID_VLD_THRESH
Specify the value for 'climo_mean.regrid.vld_thresh' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_REGRID_WIDTH
Specify the value for 'climo_mean.regrid.width' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_TIME_INTERP_METHOD
Specify the value for 'climo_mean.time_interp_method' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_USE_FCST
If set to True, use the field array from the fcst dictionary for the climo_mean fields for PointStat. Sets "climo_mean = fcst;" in the wrapped MET config file. Only used if *POINT_STAT_CLIMO_MEAN_FIELD* is unset. See also *POINT_STAT_CLIMO_MEAN_USE_OBS*.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_USE_OBS
If set to True, use the field array from the obs dictionary for the climo_mean fields for PointStat. Sets "climo_mean = obs;" in the wrapped MET config file. Only used if *POINT_STAT_CLIMO_MEAN_FIELD* is unset. See also *POINT_STAT_CLIMO_MEAN_USE_FCST*.

Used by: PointStat

POINT_STAT_CLIMO_MEAN_VAR<n>_LEVELS
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_LEVELS*

Used by: PointStat

POINT_STAT_CLIMO_MEAN_VAR<n>_NAME
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_NAME*

Used by: PointStat

POINT_STAT_CLIMO_MEAN_VAR<n>_OPTIONS
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_OPTIONS*

Used by: PointStat

POINT_STAT_CLIMO_STDEV_DAY_INTERVAL
Specify the value for 'climo_stdev.day_interval' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_FIELD
Specify the value for 'climo_stdev.field' in the MET configuration file for PointStat. The value set here must include the proper formatting that is expected in MET configuration file for specifying field information. Example: {name="TMP"; level="(,)";} To set the field information un-formatted, use the *POINT_STAT_CLIMO_STDEV_VAR<n>_NAME*, *POINT_STAT_CLIMO_STDEV_VAR<n>_LEVELS*, and *POINT_STAT_CLIMO_STDEV_VAR<n>_OPTIONS* variables.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_FILE_NAME
Specify the value for 'climo_stdev.file_name' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_HOUR_INTERVAL
Specify the value for 'climo_stdev.hour_interval' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *POINT_STAT_CLIMO_STDEV_FILE_NAME*.

POINT_STAT_CLIMO_STDEV_INPUT_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *POINT_STAT_CLIMO_STDEV_FILE_NAME*.

POINT_STAT_CLIMO_STDEV_MATCH_MONTH
Specify the value for 'climo_stdev.match_month' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_REGRID_METHOD
Specify the value for 'climo_stdev.regrid.method' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_REGRID_SHAPE
Specify the value for 'climo_stdev.regrid.shape' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_REGRID_VLD_THRESH
Specify the value for 'climo_stdev.regrid.vld_thresh' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_REGRID_WIDTH
Specify the value for 'climo_stdev.regrid.width' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_TIME_INTERP_METHOD
Specify the value for 'climo_stdev.time_interp_method' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_USE_FCST
If set to True, use the field array from the fcst dictionary for the climo_stdev fields for PointStat. Sets "climo_stdev = fcst;" in the wrapped MET config file. Only used if *POINT_STAT_CLIMO_STDEV_FIELD* is unset. See also *POINT_STAT_CLIMO_STDEV_USE_OBS*.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_USE_OBS
If set to True, use the field array from the obs dictionary for the climo_stdev fields for PointStat. Sets "climo_stdev = obs;" in the wrapped MET config file. Only used if *POINT_STAT_CLIMO_STDEV_FIELD* is unset. See also *POINT_STAT_CLIMO_STDEV_USE_FCST*.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_VAR<n>_LEVELS
Specify the level of the nth field for 'climo_stdev.field' in the MET configuration file for PointStat. If

any fields are set using this variable, then *POINT_STAT_CLIMO_STDEV_FIELD* will be ignored. See also *POINT_STAT_CLIMO_STDEV_VAR<n>_NAME* and *POINT_STAT_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_VAR<n>_NAME
Specify the name of the nth field for 'climo_stdev.field' in the MET configuration file for PointStat. If any fields are set using this variable, then *POINT_STAT_CLIMO_STDEV_FIELD* will be ignored. See also *POINT_STAT_CLIMO_STDEV_VAR<n>_LEVELS* and *POINT_STAT_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: PointStat

POINT_STAT_CLIMO_STDEV_VAR<n>_OPTIONS
Specify the extra options of the nth field for 'climo_stdev.field' in the MET configuration file for PointStat. If any fields are set using this variable, then *POINT_STAT_CLIMO_STDEV_FIELD* will be ignored. See also *POINT_STAT_CLIMO_STDEV_VAR<n>_NAME* and *POINT_STAT_CLIMO_STDEV_VAR<n>_LEVELS*.

Used by: PointStat

POINT_STAT_CONFIG_FILE
Path to configuration file read by point_stat. If unset, parm/met_config/PointStatConfig_wrapped will be used.

Used by: PointStat

POINT_STAT_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: PointStat

POINT_STAT_DESC
Specify the value for 'desc' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_FCST_FILE_TYPE
Specify the value for 'fcst.file_type' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_GRID
Specify the grid to use with the MET point_stat tool.

Note: please use *POINT_STAT_MASK_GRID*

Used by: PointStat

POINT_STAT_HIRA_COV_THRESH
Specify the value for 'hira.cov_thresh' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_HIRA_FLAG
Specify the value for 'hira.flag' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_HIRA_PROB_CAT_THRESH
Specify the value for 'hira.prob_cat_thresh' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_HIRA_SHAPE
Specify the value for 'hira.shape' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_HIRA_VLD_THRESH
Specify the value for 'hira.vld_thresh' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_HIRA_WIDTH
Specify the value for 'hira.width' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_HSS_EC_VALUE
Specify the value for 'hss_ec_value' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_INTERP_SHAPE
Specify the value for 'interp.shape' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_INTERP_TYPE_METHOD
Specify the value for 'interp.type.method' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_INTERP_TYPE_WIDTH
Specify the value for 'interp.type.width' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_INTERP_VLD_THRESH
Specify the value for 'interp.vld_thresh' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_MASK_GRID
Set the mask.grid entry in the PointStat MET config file.

Used by: PointStat

POINT_STAT_MASK_LLPNT
Specify the value for 'mask.llpnt' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_MASK_POLY
Set the mask.poly entry in the PointStat MET config file.

Used by: PointStat

POINT_STAT_MASK_SID
Set the mask.sid entry in the PointStat MET config file.

Used by: PointStat

POINT_STAT_MESSAGE_TYPE
Specify which PREPBUFR message types to process with the MET point_stat tool.

Used by: PointStat

POINT_STAT_MESSAGE_TYPE_GROUP_MAP
Specify the value for 'message_type_group_map' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: POINT_STAT_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: PointStat

POINT_STAT_NEIGHBORHOOD_SHAPE
Sets the neighborhood shape used by PointStat. See MET User's Guide for more information.

Used by: PointStat

POINT_STAT_NEIGHBORHOOD_WIDTH
Sets the neighborhood width used by PointStat. See MET User's Guide for more information.

Used by: PointStat

POINT_STAT_OBS_FILE_TYPE
Specify the value for 'obs.file_type' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OBS_QUALITY

> **Warning:** **DEPRECATED:** Please use *POINT_STAT_OBS_QUALITY_INC* instead.

POINT_STAT_OBS_QUALITY_EXC
Specify the value for 'obs_quality_exc' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OBS_QUALITY_INC
Specify the value for 'obs_quality_inc' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OBS_VALID_BEG
Optional variable that sets the -obs_valid_beg command line argument for PointStat if set to something other than an empty string. Accepts filename template syntax, i.e. {valid?fmt=%Y%m%d_%H}

Used by: PointStat

POINT_STAT_OBS_VALID_END
Optional variable that sets the -obs_valid_end command line argument for PointStat if set to something other than an empty string. Accepts filename template syntax, i.e. {valid?fmt=%Y%m%d_%H}

Used by: PointStat

POINT_STAT_OFFSETS
A list of potential offsets (in hours) that can be found in the *OBS_POINT_STAT_INPUT_TEMPLATE* and *FCST_POINT_STAT_INPUT_TEMPLATE*. METplus will check if a file with a given offset exists in the order specified in this list, to be sure to put favored offset values first.

Used by: PointStat

POINT_STAT_OUTPUT_DIR
Specify the directory where output files from the MET point_stat tool are written.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_CNT
Specify the value for 'output_flag.cnt' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_CTC
Specify the value for 'output_flag.ctc' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_CTS
Specify the value for 'output_flag.cts' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_ECLV
Specify the value for 'output_flag.eclv' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_ECNT
Specify the value for 'output_flag.ecnt' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_FHO
Specify the value for 'output_flag.fho' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_MCTC
Specify the value for 'output_flag.mctc' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_MCTS
Specify the value for 'output_flag.mcts' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_MPR
Specify the value for 'output_flag.mpr' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_ORANK
Specify the value for 'output_flag.orank' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_PCT
Specify the value for 'output_flag.pct' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_PJC
Specify the value for 'output_flag.pjc' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_PRC
Specify the value for 'output_flag.prc' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_PSTD
Specify the value for 'output_flag.pstd' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_RPS
Specify the value for 'output_flag.rps' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_SAL1L2
Specify the value for 'output_flag.sal1l2' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_SEEPS
Specify the value for 'output_flag.seeps' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_SEEPS_MPR
Specify the value for 'output_flag.seeps_mpr' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_SL1L2
Specify the value for 'output_flag.sl1l2' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_VAL1L2
Specify the value for 'output_flag.val1l2' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_VCNT
Specify the value for 'output_flag.vcnt' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_FLAG_VL1L2
Specify the value for 'output_flag.vl1l2' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_OUTPUT_PREFIX
String to pass to the MET config file to prepend text to the output filenames.

Used by: PointStat

POINT_STAT_OUTPUT_TEMPLATE
Sets the subdirectories below *POINT_STAT_OUTPUT_DIR* using a template to allow run time information. If LOOP_BY = VALID, default value is valid time YYYYMMDDHHMM/point_stat. If LOOP_BY = INIT, default value is init time YYYYMMDDHHMM/point_stat.

Used by: PointStat

POINT_STAT_POLY
Specify a polygon to use with the MET PointStat tool.

Note: please use *POINT_STAT_MASK_POLY*

Used by: PointStat

POINT_STAT_REGRID_CENSOR_THRESH
Specify the value for 'regrid.censor_thresh' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_REGRID_CENSOR_VAL
Specify the value for 'regrid.censor_val' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_REGRID_CONVERT
Specify the value for 'regrid.convert' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_REGRID_METHOD
Specify the value for 'regrid.method' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_REGRID_SHAPE
Specify the value for 'regrid.shape' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_REGRID_TO_GRID
Used to set the regrid dictionary item 'to_grid' in the MET PointStat config file. See the MET User's

Guide for more information.

Used by: PointStat

POINT_STAT_REGRID_VLD_THRESH
Specify the value for 'regrid.vld_thresh' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_REGRID_WIDTH
Specify the value for 'regrid.width' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_SEEPS_P1_THRESH
Specify the value for 'seeps_p1_thresh' in the MET configuration file for PointStat.

Used by: PointStat

POINT_STAT_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: PointStat

POINT_STAT_STATION_ID

Warning: DEPRECATED: Please use *POINT_STAT_MASK_SID* instead.

POINT_STAT_VERIFICATION_MASK_TEMPLATE
Template used to specify the verification mask filename for the MET tool point_stat. Now supports a list of filenames.

Used by: PointStat

PREFIX
This corresponds to the optional -prefix flag of the plot_TCMPR.R script (which is wrapped by TCMPRPlotter). This is the output file name prefix.

Used by: TCMPRPlotter

PREPBUFR_DATA_DIR

Warning: **DEPRECATED:** Please use *PB2NC_INPUT_DIR* instead.

PREPBUFR_DIR_REGEX

Warning: **DEPRECATED:** No longer used. Regular expression to use when searching for PREPBUFR data.

PREPBUFR_FILE_REGEX

Warning: **DEPRECATED:** No longer used. Regular expression to use when searching for PREPBUFR files.

PREPBUFR_MODEL_DIR_NAME

Warning: **DEPRECATED:** Please put the value previously used here in the *PB2NC_INPUT_DIR* path. Specify the name of the model being used with the MET pb2nc tool.

PROCESS_LIST
Specify the list of processes for METplus to perform, in a comma separated list.

Used by: All

PROJ_DIR

Warning: **DEPRECATED:** Please use *INPUT_BASE* instead.

PY_EMBED_INGEST_<n>_OUTPUT_DIR
Used to use Python embedding to process multiple files. <n> is an integer greater than or equal to 1. Specifies the output diirectory to write data. See also *PY_EMBED_INGEST_<n>_TYPE*,

PY_EMBED_INGEST_<n>_SCRIPT, and *PY_EMBED_INGEST_<n>_OUTPUT_GRID*, and *PY_EMBED_INGEST_<n>_OUTPUT_TEMPLATE*.

Used by: PyEmbedIngest

PY_EMBED_INGEST_<n>_OUTPUT_FIELD_NAME
Used to specify the forecast output field name that is created by RegridDataPlane. If this option is not set, RegridDataPlane will call the field name "name_level".

Used by: PyEmbedIngest

PY_EMBED_INGEST_<n>_OUTPUT_GRID
Used to use Python embedding to process multiple files. <n> is an integer greater than or equal to 1. Specifies the grid information that RegridDataPlane will use to generate a file that can be read by the MET tools. This can be a file path or a grid definition. See the MET User's Guide section regarding Regrid-Data-Plane for more information. See also *PY_EMBED_INGEST_<n>_TYPE*, *PY_EMBED_INGEST_<n>_SCRIPT*, *PY_EMBED_INGEST_<n>_OUTPUT_TEMPLATE*, and *PY_EMBED_INGEST_<n>_OUTPUT_DIR*.

Used by: PyEmbedIngest

PY_EMBED_INGEST_<n>_OUTPUT_TEMPLATE
Used to use Python embedding to process multiple files. <n> is an integer greater than or equal to 1. Specifies the output filename using filename template syntax. The value will be substituted with time information and appended to *PY_EMBED_INGEST_<n>_OUTPUT_DIR* if it is set. See also *PY_EMBED_INGEST_<n>_TYPE*, *PY_EMBED_INGEST_<n>_SCRIPT*, and *PY_EMBED_INGEST_<n>_OUTPUT_GRID*.

Used by: PyEmbedIngest

PY_EMBED_INGEST_<n>_SCRIPT
Used to use Python embedding to process multiple files. <n> is an integer greater than or equal to 1. Specifies the python script with arguments to run through RegridDataPlane to generate a file that can be read by the MET tools. This variable supports filename template syntax, so you can specify filenames with time information, i.e. {valid?fmt=%Y%m%d}. See also *PY_EMBED_INGEST_<n>_TYPE*, *PY_EMBED_INGEST_<n>_OUTPUT_GRID*, *PY_EMBED_INGEST_<n>_OUTPUT_TEMPLATE*, and *PY_EMBED_INGEST_<n>_OUTPUT_DIR*.

Used by: PyEmbedIngest

PY_EMBED_INGEST_<n>_TYPE
Used to use Python embedding to process multiple files. <n> is an integer greater than or equal to 1. Specifies the type of output generated by the Python script. Valid options are NUMPY, XARRAY, and PANDAS. See also *PY_EMBED_INGEST_<n>_SCRIPT*, *PY_EMBED_INGEST_<n>_OUTPUT_GRID*, *PY_EMBED_INGEST_<n>_OUTPUT_TEMPLATE*, and *PY_EMBED_INGEST_<n>_OUTPUT_DIR*.

Used by: PyEmbedIngest

PY_EMBED_INGEST_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: PyEmbedIngest

PY_EMBED_INGEST_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: PyEmbedIngest

REFERENCE_TMPL

Warning: **DEPRECATED:** Please use *TC_PAIRS_BDECK_TEMPLATE*.

REGION

Warning: **DEPRECATED:** Please use *VX_MASK_LIST* instead.

REGION_LIST

Warning: **DEPRECATED:** Please use *VX_MASK_LIST* instead.

REGRID_DATA_PLANE_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: RegridDataPlane

REGRID_DATA_PLANE_GAUSSIAN_DX
Gaussian dx value to add to the RegridDataPlane command line call with -gaussian_dx. Not added to call if unset or set to empty string.

Used by: RegridDataPlane

REGRID_DATA_PLANE_GAUSSIAN_RADIUS
Gaussian radius value to add to the RegridDataPlane command line call with -gaussian_radius. Not added to call if unset or set to empty string.

Used by: RegridDataPlane

REGRID_DATA_PLANE_METHOD
Sets the method used by regrid_data_plane. See MET User's Guide for more information.

Used by: RegridDataPlane

REGRID_DATA_PLANE_ONCE_PER_FIELD
If True, run RegridDataPlane separately for each field name/level combination specified in the configuration file. See *Field Info* (page 58) for more information on how fields are specified. If False, run RegridDataPlane once with all of the fields specified.

Used by: RegridDataPlane

REGRID_DATA_PLANE_SKIP_IF_OUTPUT_EXISTS
If True, do not run regrid_data_plane if output file already exists. Set to False to overwrite files.

Used by: RegridDataPlane

REGRID_DATA_PLANE_VERIF_GRID
Specify the absolute path to a file containing information about the desired output grid from the MET regrid_data_plane tool.

Used by: RegridDataPlane

REGRID_DATA_PLANE_WIDTH
Sets the width used by regrid_data_plane. See MET User's Guide for more information.

Used by: RegridDataPlane

REGRID_TO_GRID

Warning: **DEPRECATED:** Please use *POINT_STAT_REGRID_TO_GRID* instead.

RM

Warning: **DEPRECATED:** Do not use.

RM_EXE

Warning: **DEPRECATED:** Do not use.

RP_DIFF

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_RP_DIFF* instead.

RUN_ID

Eight character hash string unique to a given run of METplus. Automatically set by METplus at the beginning of a run. Can be referenced in other METplus config variables to distinguish multiple METplus runs that may have started within the same second. For example, it can be added to *LOG_TIMESTAMP_TEMPLATE* to create unique log files, final config files, etc.

Example: LOG_TIMESTAMP_TEMPLATE = %Y%m%d%H%M%S.{RUN_ID}

Used by: All

SAVE

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_SAVE* instead.

SAVE_DATA

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_SAVE_DATA* instead.

SCATTER_X

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_SCATTER_X* instead.

SCATTER_Y

> **Warning:** **DEPRECATED:** Please use *TCMPR_PLOTTER_SCATTER_Y* instead.

SCRUB_STAGING_DIR
If True, remove staging directory after METplus has completed running. Set to False to preserve data for subsequent runs or debugging purposes. Defaults to True.

Used by: All

SERIES

> **Warning:** **DEPRECATED:** Please use *TCMPR_PLOTTER_SERIES* instead.

SERIES_ANALYSIS_BACKGROUND_MAP
Control whether or not a background map shows up for series analysis plots. Set to 'yes' if background map desired.

Used by: SeriesAnalysis

SERIES_ANALYSIS_BLOCK_SIZE
Specify the value for 'block_size' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_BY_INIT_CONFIG_FILE

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_CONFIG_FILE* instead.

SERIES_ANALYSIS_BY_LEAD_CONFIG_FILE

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_CONFIG_FILE* instead.

SERIES_ANALYSIS_CAT_THRESH
Specify the value for 'cat_thresh' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_CDF_BINS
Specify the value for 'climo_cdf.cdf_bins' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_CDF_CENTER_BINS
Specify the value for 'climo_cdf.center_bins' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_CDF_DIRECT_PROB
Specify the value for 'climo_cdf.direct_prob' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_DAY_INTERVAL
Specify the value for 'climo_mean.day_interval' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_FIELD
See: *<TOOL-NAME>_CLIMO_MEAN_FIELD*

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_FILE_NAME
Specify the value for 'climo_mean.file_name' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_FILE_TYPE
Specify the value for 'climo_mean.file_type' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_HOUR_INTERVAL
Specify the value for 'climo_mean.hour_interval' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_CLIMO_MEAN_FILE_NAME*.

SERIES_ANALYSIS_CLIMO_MEAN_INPUT_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_CLIMO_MEAN_FILE_NAME*.

SERIES_ANALYSIS_CLIMO_MEAN_MATCH_MONTH
Specify the value for 'climo_mean.match_month' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_REGRID_METHOD
Specify the value for 'climo_mean.regrid.method' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_REGRID_SHAPE
Specify the value for 'climo_mean.regrid.shape' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_REGRID_VLD_THRESH
Specify the value for 'climo_mean.regrid.vld_thresh' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_REGRID_WIDTH
Specify the value for 'climo_mean.regrid.width' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_TIME_INTERP_METHOD
Specify the value for 'climo_mean.time_interp_method' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_USE_FCST
If set to True, use the field array from the fcst dictionary for the climo_mean fields for SeriesAnalysis. Sets "climo_mean = fcst;" in the wrapped MET config file. Only used if *SERIES_ANALYSIS_CLIMO_MEAN_FIELD* is unset. See also *SERIES_ANALYSIS_CLIMO_MEAN_USE_OBS*.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_USE_OBS
If set to True, use the field array from the obs dictionary for the climo_mean fields for SeriesAnalysis. Sets "climo_mean = obs;" in the wrapped MET config file. Only used if *SERIES_ANALYSIS_CLIMO_MEAN_FIELD* is unset. See also *SERIES_ANALYSIS_CLIMO_MEAN_USE_FCST*.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_VAR<n>_LEVELS
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_LEVELS*

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_VAR<n>_NAME
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_NAME*

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_MEAN_VAR<n>_OPTIONS
See: *<TOOL-NAME>_CLIMO_MEAN_VAR<n>_OPTIONS*

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_DAY_INTERVAL
Specify the value for 'climo_stdev.day_interval' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_FIELD
Specify the value for 'climo_stdev.field' in the MET configuration file for SeriesAnalysis. The value set here must include the proper formatting that is expected in MET configuration file for specifying field information. Example: {name="TMP"; level="(,)";} To set the field information un-formatted, use the *SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_NAME*, *SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_LEVELS*, and *SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_OPTIONS* variables.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_FILE_NAME
Specify the value for 'climo_stdev.file_name' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_FILE_TYPE
Specify the value for 'climo_stdev.file_type' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_HOUR_INTERVAL
Specify the value for 'climo_stdev.hour_interval' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR

Warning: **DEPRECATED:** Please use *SERIES_ANALYSIS_CLIMO_STDEV_FILE_NAME*.

SERIES_ANALYSIS_CLIMO_STDEV_INPUT_TEMPLATE

Warning: **DEPRECATED:** Please use *SERIES_ANALYSIS_CLIMO_STDEV_FILE_NAME*.

SERIES_ANALYSIS_CLIMO_STDEV_MATCH_MONTH
Specify the value for 'climo_stdev.match_month' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_REGRID_METHOD
Specify the value for 'climo_stdev.regrid.method' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_REGRID_SHAPE
Specify the value for 'climo_stdev.regrid.shape' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_REGRID_VLD_THRESH
Specify the value for 'climo_stdev.regrid.vld_thresh' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_REGRID_WIDTH
Specify the value for 'climo_stdev.regrid.width' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_TIME_INTERP_METHOD
Specify the value for 'climo_stdev.time_interp_method' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_USE_FCST

If set to True, use the field array from the fcst dictionary for the climo_stdev fields for SeriesAnalysis. Sets "climo_stdev = fcst;" in the wrapped MET config file. Only used if *SERIES_ANALYSIS_CLIMO_STDEV_FIELD* is unset. See also *SERIES_ANALYSIS_CLIMO_STDEV_USE_OBS*.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_USE_OBS

If set to True, use the field array from the obs dictionary for the climo_stdev fields for SeriesAnalysis. Sets "climo_stdev = obs;" in the wrapped MET config file. Only used if *SERIES_ANALYSIS_CLIMO_STDEV_FIELD* is unset. See also *SERIES_ANALYSIS_CLIMO_STDEV_USE_FCST*.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_LEVELS

Specify the level of the nth field for 'climo_stdev.field' in the MET configuration file for SeriesAnalysis. If any fields are set using this variable, then *SERIES_ANALYSIS_CLIMO_STDEV_FIELD* will be ignored. See also *SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_NAME* and *SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_NAME

Specify the name of the nth field for 'climo_stdev.field' in the MET configuration file for SeriesAnalysis. If any fields are set using this variable, then *SERIES_ANALYSIS_CLIMO_STDEV_FIELD* will be ignored. See also *SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_LEVELS* and *SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_OPTIONS*.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_OPTIONS

Specify the extra options of the nth field for 'climo_stdev.field' in the MET configuration file for SeriesAnalysis. If any fields are set using this variable, then *SERIES_ANALYSIS_CLIMO_STDEV_FIELD* will be ignored. See also *SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_NAME* and *SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_LEVELS*.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CONFIG_FILE
Path to configuration file read by series_analysis. If unset, parm/met_config/SeriesAnalysisConfig_wrapped will be used.

Used by: SeriesAnalysis

SERIES_ANALYSIS_CTS_LIST

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_OUTPUT_STATS_CTS* instead.

SERIES_ANALYSIS_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: SeriesAnalysis

SERIES_ANALYSIS_DESC
Specify the value for 'desc' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_FILTER_OPTS

> **Warning:** **DEPRECATED:** Please use *TC_STAT_JOB_ARGS* instead.

SERIES_ANALYSIS_FILTERED_OUTPUT

> **Warning:** **DEPRECATED:** No longer used.

SERIES_ANALYSIS_FILTERED_OUTPUT_DIR

> **Warning:** **DEPRECATED:** No longer used.

SERIES_ANALYSIS_GENERATE_ANIMATIONS
If set to True, create GIF animated images images. Previously, animated images were always generated.

Used by: SeriesAnalysis

SERIES_ANALYSIS_GENERATE_PLOTS
If set to True, run plot_data_plane and convert to generate images. Previously, plots were always generated.

Used by: SeriesAnalysis

SERIES_ANALYSIS_GROUP_FCSTS

> **Warning:** **DEPRECATED:** Please use *LEAD_SEQ_<n>* and *SERIES_ANALYSIS_RUNTIME_FREQ* instead.

SERIES_ANALYSIS_HSS_EC_VALUE
Specify the value for 'hss_ec_value' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_TILE_INPUT_DIR* instead.

SERIES_ANALYSIS_IS_PAIRED
If true, the -paired flag is added to the SeriesAnalysis command.

Used by: SeriesAnalysis

SERIES_ANALYSIS_MASK_GRID
Specify the value for 'mask.grid' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_MASK_POLY
Specify the value for 'mask.poly' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_MET_CONFIG_OVERRIDES

Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: SERIES_ANALYSIS_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_DIR

Specify the directory where files will be written from the MET series analysis tool.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_CNT

Specify the value for 'output_stats.cnt' in the MET configuration file for SeriesAnalysis. Also used to generate plots for each value in the list.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_CTC

Specify the value for 'output_stats.ctc' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_CTS

Specify the value for 'output_stats.cts' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_FHO

Specify the value for 'output_stats.fho' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_MCTC
Specify the value for 'output_stats.mctc' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_MCTS
Specify the value for 'output_stats.mcts' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_PCT
Specify the value for 'output_stats.pct' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_PJC
Specify the value for 'output_stats.pjc' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_PRC
Specify the value for 'output_stats.prc' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_PSTD
Specify the value for 'output_stats.pstd' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_SAL1L2
Specify the value for 'output_stats.sal1l2' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_STATS_SL1L2
Specify the value for 'output_stats.sl1l2' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_OUTPUT_TEMPLATE
Filename template of the output file generated by SeriesAnalysis. See also *SERIES_ANALYSIS_OUTPUT_DIR*.

Used by: SeriesAnalysis

SERIES_ANALYSIS_REGRID_CENSOR_THRESH
Specify the value for 'regrid.censor_thresh' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_REGRID_CENSOR_VAL
Specify the value for 'regrid.censor_val' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_REGRID_CONVERT
Specify the value for 'regrid.convert' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_REGRID_METHOD
Specify the value for 'regrid.method' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_REGRID_SHAPE
Specify the value for 'regrid.shape' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_REGRID_TO_GRID
Used to set the regrid dictionary item 'to_grid' in the MET SeriesAnalysis config file. See the MET User's Guide for more information.

Used by: SeriesAnalysis

SERIES_ANALYSIS_REGRID_VLD_THRESH
Specify the value for 'regrid.vld_thresh' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_REGRID_WIDTH
Specify the value for 'regrid.width' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID
If True, run SeriesAnalysis once for each storm ID found in the .tcst (TCStat output) file specified with *SERIES_ANALYSIS_TC_STAT_INPUT_DIR* and *SERIES_ANALYSIS_TC_STAT_INPUT_TEMPLATE*.

Used by: SeriesAnalysis

SERIES_ANALYSIS_RUNTIME_FREQ
Frequency to run SeriesAnalysis. See *Runtime Frequency* (page 70) for more information.

Used by: SeriesAnalysis

SERIES_ANALYSIS_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: SeriesAnalysis

SERIES_ANALYSIS_STAT_INPUT_DIR

Warning: **DEPRECATED:** Please use *SERIES_ANALYSIS_TC_STAT_INPUT_DIR* instead.

SERIES_ANALYSIS_STAT_INPUT_TEMPLATE

Warning: **DEPRECATED:** Please use *SERIES_ANALYSIS_TC_STAT_INPUT_TEMPLATE* instead.

SERIES_ANALYSIS_STAT_LIST

Warning: **DEPRECATED:** Please use *SERIES_ANALYSIS_OUTPUT_STATS_CNT* instead.

SERIES_ANALYSIS_TC_STAT_INPUT_DIR
Directory containing TCStat output to be read by SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_ANALYSIS_TC_STAT_INPUT_TEMPLATE
Template used to specify the dump row output tcst file generated by TCStat to filter input data to be used in SeriesAnalysis. Example: {init?fmt=%Y%m%d_%H}/filter_{init?fmt=%Y%m%d_%H}.tcst

Used by: SeriesAnalysis

SERIES_ANALYSIS_TILE_INPUT_DIR

Warning: **DEPRECATED:** Please use *FCST_SERIES_ANALYSIS_INPUT_DIR* and *OBS_SERIES_ANALYSIS_INPUT_DIR* instead.

SERIES_ANALYSIS_VAR_LIST

Warning: **DEPRECATED:** Please use *FCST_VAR<n>_NAME* and *OBS_VAR<n>_NAME* instead.

SERIES_ANALYSIS_VLD_THRESH
Specify the value for 'vld_thresh' in the MET configuration file for SeriesAnalysis.

Used by: SeriesAnalysis

SERIES_BY_INIT_FILTERED_OUTPUT_DIR

> **Warning:** **DEPRECATED:** No longer used.

SERIES_BY_INIT_OUTPUT_DIR

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_OUTPUT_DIR* instead.

SERIES_BY_LEAD_FILTERED_OUTPUT

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_FILTERED_OUTPUT_DIR* instead.

SERIES_BY_LEAD_FILTERED_OUTPUT_DIR

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_FILTERED_OUTPUT_DIR* instead.

SERIES_BY_LEAD_GROUP_FCSTS

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_GROUP_FCSTS* instead.

SERIES_BY_LEAD_OUTPUT_DIR

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_OUTPUT_DIR* instead.

SERIES_CI

> **Warning:** **DEPRECATED:** Please use *TCMPR_PLOTTER_SERIES_CI* instead.

SERIES_INIT_FILTERED_OUT_DIR

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_FILTERED_OUTPUT_DIR* instead.

SERIES_INIT_OUT_DIR

Warning: **DEPRECATED:** Please use *SERIES_ANALYSIS_OUTPUT_DIR* instead.

SERIES_LEAD_FILTERED_OUT_DIR

Warning: **DEPRECATED:** Please use *SERIES_ANALYSIS_FILTERED_OUTPUT_DIR*.

SERIES_LEAD_OUT_DIR

Warning: **DEPRECATED:** Please use *SERIES_ANALYSIS_OUTPUT_DIR* instead.

SKILL_REF

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_SKILL_REF* instead.

SKIP_TIMES

List of valid times to skip processing. Each value be surrounded by quotation marks and must contain a datetime format followed by a list of matching times to skip. Multiple items can be defined separated by commas. begin_end_incr syntax can be used to define a list as well.

Examples:

Value: SKIP_TIMES = "%m:11,12"

Result: Skip the 11th and 12th month

Value: SKIP_TIMES = "%m:11", "%d:31"

Result: Skip if 11th month or 31st day.

Value: SKIP_TIMES = "%Y%m%d:20201031"

Result: Skip October 31, 2020

Value: SKIP_TIMES = "%H:begin_end_incr(0,22, 2)"

Result: Skip even hours: 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22

Used by: GridStat, SeriesAnalysis

STAGING_DIR

Directory to store intermediate files such as data files that were automatically uncompressed or converted. Also includes *FILE_LISTS_DIR* by default.

Used by: All

START_DATE

> **Warning:** **DEPRECATED:** Please use *INIT_BEG* or *VALID_BEG* instead.

START_HOUR

> **Warning:** **DEPRECATED:** Please use *INIT_BEG* or *VALID_BEG* instead.

STAT_ANALYSIS_CONFIG

> **Warning:** **DEPRECATED:** Please use *STAT_ANALYSIS_CONFIG_FILE* instead.

STAT_ANALYSIS_CONFIG_FILE
Path to optional configuration file read by stat_analysis. To utilize a configuration file, set this to {PARM_BASE}/parm/met_config/STATAnalysisConfig_wrapped. If unset, no config file will be used.

Used by: StatAnalysis

STAT_ANALYSIS_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: StatAnalysis

STAT_ANALYSIS_DUMP_ROW_TMPL

> **Warning:** **DEPRECATED:** Please use *MODEL<n>_STAT_ANALYSIS_DUMP_ROW_TEMPLATE* instead.

STAT_ANALYSIS_FCST_INIT_BEG
Specify the value for 'fcst_init_beg' in the MET configuration file for StatAnalysis. This can refer to filename template tags that are set by the wrapper. Example:

```
[config]
INIT_BEG = 20221014
STAT_ANALYSIS_FCST_INIT_BEG = {fcst_init_beg?fmt=%Y%m%d_%H}
```

will set fcst_init_beg = "20221014_00"; in the wrapped MET config file.

Used by: StatAnalysis

STAT_ANALYSIS_FCST_INIT_END

Specify the value for 'fcst_init_end' in the MET configuration file for StatAnalysis. This can refer to filename template tags that are set by the wrapper. Example:

```
[config]
INIT_END = 20221015
STAT_ANALYSIS_FCST_INIT_END = {fcst_init_beg?fmt=%Y%m%d}_12
```

will set fcst_init_end = "20221014_12"; in the wrapped MET config file.

Used by: StatAnalysis

STAT_ANALYSIS_FCST_VALID_BEG

Specify the value for 'fcst_valid_beg' in the MET configuration file for StatAnalysis. This can refer to filename template tags that are set by the wrapper. Example:

```
[config]
VALID_BEG = 20221014
STAT_ANALYSIS_FCST_VALID_BEG = {fcst_valid_beg?fmt=%Y%m%d_%H}
```

will set fcst_valid_beg = "20221014_00"; in the wrapped MET config file.

Used by: StatAnalysis

STAT_ANALYSIS_FCST_VALID_END

Specify the value for 'fcst_valid_end' in the MET configuration file for StatAnalysis. This can refer to filename template tags that are set by the wrapper. Example:

```
[config]
VALID_END = 20221015
STAT_ANALYSIS_FCST_VALID_END = {fcst_valid_beg?fmt=%Y%m%d}_12
```

will set fcst_valid_end = "20221014_12"; in the wrapped MET config file.

Used by: StatAnalysis

STAT_ANALYSIS_HSS_EC_VALUE

Specify the value for 'hss_ec_value' in the MET configuration file for StatAnalysis.

Used by: StatAnalysis

STAT_ANALYSIS_INIT_BEG

Specify the value for both 'fcst_init_beg' and 'obs_init_beg' in the MET configuration file for StatAnalysis. See *STAT_ANALYSIS_FCST_INIT_BEG*.

Used by: StatAnalysis

STAT_ANALYSIS_INIT_END

Specify the value for both 'fcst_init_end' and 'obs_init_end' in the MET configuration file for StatAnalysis. See *STAT_ANALYSIS_FCST_INIT_END*.

Used by: StatAnalysis

STAT_ANALYSIS_JOB<n>

Specify StatAnalysis job arguments to run. Include the full set of job arguments including the -job argument. Multiple jobs can be defined by with STAT_ANALYSIS_JOB1, STAT_ANALYSIS_JOB2, etc. Filename template tags can be used to insert values from a given run into the job arguments. The keywords [dump_row_file] and [out_stat_file] can be used and will be substituted with values from *MODEL<n>_STAT_ANALYSIS_DUMP_ROW_TEMPLATE* and *MODEL<n>_STAT_ANALYSIS_OUT_STAT_TEMPLATE* respectively.

Used by: StatAnalysis

STAT_ANALYSIS_JOB_ARGS

> **Warning:** **DEPRECATED:** Please use *STAT_ANALYSIS_JOB<n>* instead.

STAT_ANALYSIS_JOB_NAME

> **Warning:** **DEPRECATED:** Please use *STAT_ANALYSIS_JOB<n>* instead.

STAT_ANALYSIS_LOOKIN_DIR

> **Warning:** **DEPRECATED:** Please use *MODEL<n>_STAT_ANALYSIS_LOOKIN_DIR* instead.

STAT_ANALYSIS_MET_CONFIG_OVERRIDES

Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: STAT_ANALYSIS_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: StatAnalysis

STAT_ANALYSIS_OBS_INIT_BEG

Specify the value for 'obs_init_beg' in the MET configuration file for StatAnalysis. This can refer to filename template tags that are set by the wrapper. Example:

```
[config]
INIT_BEG = 20221014
STAT_ANALYSIS_OBS_INIT_BEG = {obs_init_beg?fmt=%Y%m%d_%H}
```

will set obs_init_beg = "20221014_00"; in the wrapped MET config file.

Used by: StatAnalysis

STAT_ANALYSIS_OBS_INIT_END

Specify the value for 'obs_init_end' in the MET configuration file for StatAnalysis. This can refer to filename template tags that are set by the wrapper. Example:

```
[config]
INIT_END = 20221015
STAT_ANALYSIS_OBS_INIT_END = {obs_init_end?fmt=%Y%m%d}_12
```

will set obs_init_end = "20221014_12"; in the wrapped MET config file.

Used by: StatAnalysis

STAT_ANALYSIS_OBS_VALID_BEG

Specify the value for 'obs_valid_beg' in the MET configuration file for StatAnalysis. This can refer to filename template tags that are set by the wrapper. Example:

```
[config]
VALID_BEG = 20221014
STAT_ANALYSIS_OBS_VALID_BEG = {obs_valid_beg?fmt=%Y%m%d_%H}
```

will set obs_valid_beg = "20221014_00"; in the wrapped MET config file.

Used by: StatAnalysis

STAT_ANALYSIS_OBS_VALID_END

Specify the value for 'obs_valid_end' in the MET configuration file for StatAnalysis. This can refer to filename template tags that are set by the wrapper. Example:

```
[config]
VALID_END = 20221015
STAT_ANALYSIS_OBS_VALID_END = {obs_valid_end?fmt=%Y%m%d}_12
```

will set obs_valid_end = "20221014_12"; in the wrapped MET config file.

Used by: StatAnalysis

STAT_ANALYSIS_OUT_DIR

Warning: **DEPRECATED:** Please use *STAT_ANALYSIS_OUTPUT_DIR* instead.

STAT_ANALYSIS_OUT_STAT_TMPL

Warning: **DEPRECATED:** Please use *MODEL<n>_STAT_ANALYSIS_OUT_STAT_TEMPLATE* instead.

STAT_ANALYSIS_OUTPUT_DIR

This is the base directory where the output from running stat_analysis_wrapper will be put.

Used by: StatAnalysis

STAT_ANALYSIS_OUTPUT_TEMPLATE

(Optional) Specify the template of the output file to write job output from stat_analysis. If set, then the -out command line argument with the full path to the file will be added to the stat_analysis call.

Used by: StatAnalysis

STAT_ANALYSIS_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: StatAnalysis

STAT_ANALYSIS_VALID_BEG
Specify the value for both 'fcst_valid_beg' and 'obs_valid_beg' in the MET configuration file for StatAnalysis. See *STAT_ANALYSIS_FCST_VALID_BEG*.

Used by: StatAnalysis

STAT_ANALYSIS_VALID_END
Specify the value for both 'fcst_valid_end' and 'obs_valid_end' in the MET configuration file for StatAnalysis. See *STAT_ANALYSIS_FCST_VALID_END*.

Used by: StatAnalysis

STAT_LIST

> **Warning:** **DEPRECATED:** Please use *SERIES_ANALYSIS_OUTPUT_STATS_CNT* instead.

STORM_ID

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_STORM_ID* or *TC_STAT_STORM_ID*.

STORM_NAME

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_STORM_NAME*.

SUBTITLE

> **Warning:** **DEPRECATED:** Please use *TCMPR_PLOTTER_SUBTITLE*.

TC_DIAG_BASIN
Specify the value for 'basin' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_CENSOR_THRESH
Specify the value for 'censor_thresh' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_CENSOR_VAL
Specify the value for 'censor_val' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_CIRA_DIAG_FLAG
Specify the value for 'cira_diag_flag' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_COMPUTE_TANGENTIAL_AND_RADIAL_WINDS
Specify the value for 'compute_tangential_and_radial_winds' in the MET configuration file for TCDiag .

Used by: TCDiag

TC_DIAG_CONFIG_FILE
Path to configuration file read by tc_diag. If unset, parm/met_config/TCDiagConfig_wrapped will be used.

Used by: TCDiag

TC_DIAG_CONVERT
Specify the value for 'convert' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_CYCLONE
Specify the value for 'cyclone' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_DATA_DOMAIN
Specify the value for 'data.domain' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_DATA_LEVEL
Specify the value for 'data.level' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_DECK_INPUT_DIR
Directory containing ADECK input data to TCDiag. This variable is optional because you can specify the full path to the input files using *TC_DIAG_DECK_TEMPLATE*.

Used by: TCDiag

TC_DIAG_DECK_TEMPLATE
Filename template of the ADECK input data used by TCDiag. See also *TC_DIAG_DECK_INPUT_DIR*.

Used by: TCDiag

TC_DIAG_DIAG_SCRIPT
Specify the value for 'diag_script' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_DOMAIN_INFO<n>_DELTA_RANGE_KM
Specify the value for the nth 'domain_info.delta_range_km' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_DOMAIN_INFO<n>_DIAG_SCRIPT
Specify the value for the nth 'domain_info.diag_script' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_DOMAIN_INFO<n>_DOMAIN
Specify the value for the nth 'domain_info.domain' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_DOMAIN_INFO<n>_N_AZIMUTH
Specify the value for the nth 'domain_info.n_azimuth' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_DOMAIN_INFO<n>_N_RANGE
Specify the value for the nth 'domain_info.n_range' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_INIT_INCLUDE
Specify the value for 'init_inc' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_INPUT_DATATYPE
Specify the data type of the input directory for input files used with the MET tc_diag tool. Used to set the 'file_type' value of the data dictionary in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_INPUT_DIR
Directory containing input data to TCDiag. This variable is optional because you can specify the full path to the input files using *TC_DIAG_INPUT_TEMPLATE*.

Used by: TCDiag

TC_DIAG_INPUT_FILE_LIST
Specifies an explicit path to a file list file to pass into tc_diag. If set, *TC_DIAG_INPUT_TEMPLATE* and *TC_DIAG_INPUT_DIR* are ignored.

Used by: TCDiag

TC_DIAG_INPUT_TEMPLATE
Filename template of the input data used by TCDiag. See also *TC_DIAG_INPUT_DIR*.

Used by: TCDiag

TC_DIAG_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: TC_DIAG_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: TCDiag

TC_DIAG_MODEL
Specify the value for 'model' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_NC_DIAG_FLAG
Specify the value for 'nc_diag_flag' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_NC_RNG_AZI_FLAG
Specify the value for 'nc_rng_azi_flag' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_OUTPUT_DIR
Directory to write output data from TCDiag. This variable is optional because you can specify the full path to the output file using *TC_DIAG_OUTPUT_TEMPLATE*.

Used by: TCDiag

TC_DIAG_OUTPUT_PREFIX
Specify the value for 'output_prefix' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_OUTPUT_TEMPLATE
Filename template of write the output data generated by TCDiag. See also *TC_DIAG_OUTPUT_DIR*.

Used by: TCDiag

TC_DIAG_RADIAL_VELOCITY_FIELD_NAME
Specify the value for 'radial_velocity_field_name' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_RADIAL_VELOCITY_LONG_FIELD_NAME
Specify the value for 'radial_velocity_long_field_name' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_REGRID_CENSOR_THRESH
Specify the value for 'regrid.censor_thresh' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_REGRID_CENSOR_VAL
Specify the value for 'regrid.censor_val' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_REGRID_CONVERT
Specify the value for 'regrid.convert' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_REGRID_METHOD
Specify the value for 'regrid.method' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_REGRID_SHAPE
Specify the value for 'regrid.shape' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_REGRID_VLD_THRESH
Specify the value for 'regrid.vld_thresh' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_REGRID_WIDTH
Specify the value for 'regrid.width' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: TCDiag

TC_DIAG_STORM_ID
Specify the value for 'storm_id' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_TANGENTIAL_VELOCITY_FIELD_NAME
Specify the value for 'tangential_velocity_field_name' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_TANGENTIAL_VELOCITY_LONG_FIELD_NAME
Specify the value for 'tangential_velocity_long_field_name' in the MET configuration file for TCDiag .

Used by: TCDiag

TC_DIAG_U_WIND_FIELD_NAME
Specify the value for 'u_wind_field_name' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_V_WIND_FIELD_NAME
Specify the value for 'v_wind_field_name' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_VALID_BEG
Specify the value for 'valid_beg' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_VALID_END
Specify the value for 'valid_end' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_VALID_EXCLUDE
Specify the value for 'valid_exc' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_VALID_HOUR
Specify the value for 'valid_hour' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_VALID_INCLUDE
Specify the value for 'valid_inc' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_DIAG_VORTEX_REMOVAL
Specify the value for 'vortex_removal' in the MET configuration file for TCDiag.

Used by: TCDiag

TC_GEN_BASIN_FILE
Specify the value of 'basin_file' in the MET configuration file.

Used by: TCGen

TC_GEN_BASIN_MASK
Specify the 'basin_mask' value to set in the MET configuration file.

Used by: TCGen

TC_GEN_BEST_GENESIS_CATEGORY
Specify the value of best_genesis.category in the MET configuration file.

Used by: TCGen

TC_GEN_BEST_GENESIS_MSLP_THRESH
Specify the value of best_genesis.mslp_thresh in the MET configuration file.

Used by: TCGen

TC_GEN_BEST_GENESIS_TECHNIQUE
Specify the value of best_genesis.technique in the MET configuration file.

Used by: TCGen

TC_GEN_BEST_GENESIS_VMAX_THRESH
Specify the value of best_genesis.vmax_thresh in the MET configuration file.

Used by: TCGen

TC_GEN_BEST_UNIQUE_FLAG
Specify the value of 'best_unique_flag' in the MET configuration file.

Used by: TCGen

TC_GEN_CI_ALPHA
Specify the value of 'ci_alpha' in the MET configuration file.

Used by: TCGen

TC_GEN_CONFIG_FILE
Path to configuration file read by tc_gen. If unset, parm/met_config/TCGenConfig_wrapped will be used.

Used by: TCGen

TC_GEN_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: TCGen

TC_GEN_DESC
Specify the value for 'desc' in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_DEV_HIT_RADIUS
Specify the value of 'dev_hit_radius' in the MET configuration file.

Used by: TCGen

TC_GEN_DEV_HIT_WINDOW_BEGIN
Specify the value for dev_hit_window.begin in the MET configuration file.

Used by: TCGen

TC_GEN_DEV_HIT_WINDOW_END
Specify the value of dev_hit_window.end in the MET configuration file.

Used by: TCGen

TC_GEN_DEV_METHOD_FLAG
Specify the value of 'dev_method_flag' in the MET configuration file.

Used by: TCGen

TC_GEN_DISCARD_INIT_POST_GENESIS_FLAG
Specify the value of 'discard_init_post_genesis_flag' in the MET configuration file.

Used by: TCGen

TC_GEN_DLAND_FILE
Specify the value of 'dland_file' in the MET configuration file.

Used by: TCGen

TC_GEN_DLAND_THRESH
Specify the value of 'dland_thresh' in the MET configuration file.

Used by: TCGen

TC_GEN_EDECK_INPUT_DIR
Directory containing the edeck data used by TCGen. This variable is optional because you can specify the full path to edeck data using *TC_GEN_EDECK_INPUT_TEMPLATE*.

Used by: TCGen

TC_GEN_EDECK_INPUT_TEMPLATE
Filename template of the edeck data used by TCGen. See also *TC_GEN_EDECK_INPUT_DIR*.

Used by: TCGen

TC_GEN_FCST_GENESIS_MSLP_THRESH
Specify the value of fcst_genesis.mslp_thresh in the MET configuration file.

Used by: TCGen

TC_GEN_FCST_GENESIS_VMAX_THRESH
Specify the value of fcst_genesis.vmax_thresh in the MET configuration file.

Used by: TCGen

TC_GEN_FCST_HR_WINDOW_BEGIN
Specify the value of fcst_hr_window.begin in the MET configuration file.

Used by: TCGen

TC_GEN_FCST_HR_WINDOW_END
Specify the value of fcst_hr_window.end in the MET configuration file.

Used by: TCGen

TC_GEN_FILTER_<n>
Specify the values of 'filter' in the MET configuration file where <n> is any integer. Any quotation marks that are found inside another set of quotation marks must be preceded with a backslash

Used by: TCGen

TC_GEN_GENESIS_INPUT_DIR
Directory containing the genesis data used by TCGen. This variable is optional because you can specify the full path to genesis data using *TC_GEN_GENESIS_INPUT_TEMPLATE*.

Used by: TCGen

TC_GEN_GENESIS_INPUT_TEMPLATE
Filename template of the genesis data used by TCGen. See also *TC_GEN_GENESIS_INPUT_DIR*.

Used by: TCGen

TC_GEN_GENESIS_MATCH_POINT_TO_TRACK
Specify the value for 'genesis_match_point_to_track' in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_GENESIS_MATCH_RADIUS
Specify the value of 'genesis_match_radius' in the MET configuration file.

Used by: TCGen

TC_GEN_GENESIS_MATCH_WINDOW_BEG
Specify the value for 'genesis_match_window.beg' in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_GENESIS_MATCH_WINDOW_END
Specify the value for 'genesis_match_window.end' in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_GENESIS_RADIUS

> **Warning:** **DEPRECATED:** Please use *TC_GEN_GENESIS_MATCH_RADIUS* and *TC_GEN_DEV_HIT_RADIUS*.

TC_GEN_GENESIS_WINDOW_BEGIN

> **Warning:** **DEPRECATED:** Please use *TC_GEN_DEV_HIT_WINDOW_BEGIN*.

TC_GEN_GENESIS_WINDOW_END

> **Warning:** **DEPRECATED:** Please use *TC_GEN_DEV_HIT_WINDOW_END*.

TC_GEN_INIT_BEG
Specify the beginning initialization time for stratification when using the MET TCGen tool. Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCGen

TC_GEN_INIT_END
Specify the ending initialization time for stratification when using the MET TCGen tool. Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCGen

TC_GEN_INIT_EXC
Specify the value of 'init_exc' in the MET configuration file.

Used by: TCGen

TC_GEN_INIT_FREQ
Specify the value of 'init_freq' in the MET configuration file.

Used by: TCGen

TC_GEN_INIT_HOUR
Specify a list of hours for initialization times for use in the analysis.

Used by: TCGen

TC_GEN_INIT_INC
Specify the value of 'init_inc' in the MET configuration file.

Used by: TCGen

TC_GEN_LEAD_WINDOW_BEGIN

> **Warning:** **DEPRECATED:** Please use *TC_GEN_FCST_HR_WINDOW_BEGIN*.

TC_GEN_LEAD_WINDOW_END

> **Warning:** **DEPRECATED:** Please use *TC_GEN_FCST_HR_WINDOW_END*.

TC_GEN_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: TC_GEN_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: TCGen

TC_GEN_MIN_DURATION
Specify the value of 'min_duration' in the MET configuration file.

Used by: TCGen

TC_GEN_NC_PAIRS_FLAG_BEST_FN_OY
Specify the value of nc_pairs_flag.best_fn_oy in the MET configuration file.

Used by: TCGen

TC_GEN_NC_PAIRS_FLAG_BEST_FY_OY
Specify the value of nc_pairs_flag.best_fy_oy in the MET configuration file.

Used by: TCGen

TC_GEN_NC_PAIRS_FLAG_BEST_GENESIS
Specify the value of nc_pairs_flag.best_genesis in the MET configuration file.

Used by: TCGen

TC_GEN_NC_PAIRS_FLAG_BEST_TRACKS
Specify the value of nc_pairs_flag.best_tracks in the MET configuration file.

Used by: TCGen

TC_GEN_NC_PAIRS_FLAG_FCST_FY_ON
Specify the value of nc_pairs_flag.fcst_fy_on in the MET configuration file.

Used by: TCGen

TC_GEN_NC_PAIRS_FLAG_FCST_FY_OY
Specify the value of nc_pairs_flag.fcst_fy_oy in the MET configuration file.

Used by: TCGen

TC_GEN_NC_PAIRS_FLAG_FCST_GENESIS
Specify the value of nc_pairs_flag.fcst_genesis in the MET configuration file.

Used by: TCGen

TC_GEN_NC_PAIRS_FLAG_FCST_TRACKS
Specify the value of nc_pairs_flag.fcst_tracks in the MET configuration file.

Used by: TCGen

TC_GEN_NC_PAIRS_FLAG_LATLON
Specify the value of nc_pairs_flag.latlon in the MET configuration file.

Used by: TCGen

TC_GEN_NC_PAIRS_GRID
Specify the value of 'nc_pairs_grid' in the MET configuration file.

Used by: TCGen

TC_GEN_OPER_GENESIS_CATEGORY

> **Warning:** **DEPRECATED:** Please use *TC_GEN_OPER_TECHNIQUE*.

TC_GEN_OPER_GENESIS_MSLP_THRESH

> **Warning:** **DEPRECATED:** Please use *TC_GEN_OPER_TECHNIQUE*.

TC_GEN_OPER_GENESIS_TECHNIQUE

> **Warning:** **DEPRECATED:** Please use *TC_GEN_OPER_TECHNIQUE*.

TC_GEN_OPER_GENESIS_VMAX_THRESH

> **Warning:** **DEPRECATED:** Please use *TC_GEN_OPER_TECHNIQUE*.

TC_GEN_OPER_TECHNIQUE
Specify the value of 'oper_technique' in the MET configuration file.

Used by: TCGen

TC_GEN_OPS_HIT_WINDOW_BEG
Specify the value for 'ops_hit_window.beg' in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_OPS_HIT_WINDOW_END
Specify the value for 'ops_hit_window.end' in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_OPS_METHOD_FLAG
Specify the value of 'ops_method_flag' in the MET configuration file.

Used by: TCGen

TC_GEN_OUTPUT_DIR
Specify the output directory where files from the MET TCGen tool are written.

Used by: TCGen

TC_GEN_OUTPUT_FLAG_CTC
Specify the value of output_flag.ctc in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_OUTPUT_FLAG_CTS
Specify the value of output_flag.cts in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_OUTPUT_FLAG_FHO
Specify the value of output_flag.fho in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_OUTPUT_FLAG_GENMPR
Specify the value of output_flag.genmpr in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_OUTPUT_FLAG_PCT
Specify the value for 'output_flag.pct' in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_OUTPUT_FLAG_PJC
Specify the value for 'output_flag.pjc' in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_OUTPUT_FLAG_PRC
Specify the value for 'output_flag.prc' in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_OUTPUT_FLAG_PSTD
Specify the value for 'output_flag.pstd' in the MET configuration file for TCGen.

Used by: TCGen

TC_GEN_OUTPUT_TEMPLATE
Sets the subdirectories below *TC_GEN_OUTPUT_DIR* using a template to allow run time information.

Used by: TCGen

TC_GEN_SHAPE_INPUT_DIR
Directory containing the shape data used by TCGen. This variable is optional because you can specify the full path to shape data using *TC_GEN_SHAPE_INPUT_TEMPLATE*.

Used by: TCGen

TC_GEN_SHAPE_INPUT_TEMPLATE
Filename template of the shape data used by TCGen. See also *TC_GEN_SHAPE_INPUT_DIR*.

Used by: TCGen

TC_GEN_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: TCGen

TC_GEN_STORM_ID
The identifier of the storm(s) of interest.

Used by: TCGen

TC_GEN_STORM_NAME
The name(s) of the storm of interest.

Used by: TCGen

TC_GEN_TRACK_INPUT_DIR
Directory containing the track data used by TCGen. This variable is optional because you can specify the full path to track data using *TC_GEN_TRACK_INPUT_TEMPLATE*.

Used by: TCGen

TC_GEN_TRACK_INPUT_TEMPLATE
Filename template of the track data used by TCGen. See also *TC_GEN_TRACK_INPUT_DIR*.

Used by: TCGen

TC_GEN_VALID_BEG
Specify the beginning valid time for stratification when using the MET TCGen tool. Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCGen

TC_GEN_VALID_END
Specify the ending valid time for stratification when using the MET TCGen tool. Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCGen

TC_GEN_VALID_FREQ
Specify the value of 'valid_freq' in the MET configuration file.

Used by: TCGen

TC_GEN_VALID_MINUS_GENESIS_DIFF_THRESH
Specify the value of 'valid_minus_genesis_diff_thresh' in the MET configuration file.

Used by: TCGen

TC_GEN_VX_MASK
Specify the 'vx_mask' value to set in the MET configuration file.

Used by: TCGen

TC_PAIRS_ADECK_INPUT_DIR
Directory that contains the ADECK files.

Used by: TCPairs

TC_PAIRS_ADECK_INPUT_TEMPLATE
Template of the file names of ADECK data.

Used by: TCPairs

TC_PAIRS_ADECK_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_ADECK_INPUT_TEMPLATE*.

TC_PAIRS_BASIN
Control what basins are desired for tropical cyclone analysis. Per the MET User's Guide acceptable

basin ID's are:WP = Western Northern PacificIO = Northern Indian OceanSH = Southern HemisphereCP = Central Northern PacificEP = Eastern Northern PacificAL = Northern AtlanticSL = Southern Atlantic

Used by: TCPairs

TC_PAIRS_BDECK_INPUT_DIR
Directory that contains the BDECK files.

Used by: TCPairs

TC_PAIRS_BDECK_INPUT_TEMPLATE
Template of the file names of BDECK data.

Used by: TCPairs

TC_PAIRS_BDECK_TEMPLATE

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_BDECK_INPUT_TEMPLATE*.

TC_PAIRS_CHECK_DUP
Specify the value for 'check_dup' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_CONFIG_FILE
Path to configuration file read by tc_pairs. If unset, parm/met_config/TCPairsConfig_wrapped will be used.

Used by: TCPairs

TC_PAIRS_CONSENSUS<n>_MEMBERS
Specify the value for the nth 'consensus.members' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_CONSENSUS<n>_MIN_REQ
Specify the value for the nth 'consensus.min_req' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_CONSENSUS<n>_NAME
Specify the value for the nth 'consensus.name' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_CONSENSUS<n>_REQUIRED
Specify the value for the nth 'consensus.required' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_CONSENSUS<n>_WRITE_MEMBERS
Specify the value for the nth 'consensus.write_members' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_CUSTOM_LOOP_LIST
Sets custom string loop list for a specific wrapper. See *CUSTOM_LOOP_LIST*.

Used by: TCPairs

TC_PAIRS_CYCLONE
Specify which cyclone numbers to include in the tropical cyclone analysis. Per the MET User's Guide, this can be any number 01-99 (HH format). Use a space or comma separated list, or leave unset if all cyclones are desired.

Used by: TCPairs

TC_PAIRS_DESC
Specify the value for 'desc' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_DIAG_CONVERT_MAP<n>_CONVERT
Specify the value for the nth 'diag_convert_map.convert' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_DIAG_CONVERT_MAP<n>_DIAG_SOURCE
Specify the value for the nth 'diag_convert_map.diag_source' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_DIAG_CONVERT_MAP<n>_KEY
Specify the value for the nth 'diag_convert_map.key' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_DIAG_DIR<n>
Specify the (optional) directory for the nth -diag argument for TCPairs.

Used by: TCPairs

TC_PAIRS_DIAG_INFO_MAP<n>_DIAG_NAME
Specify the value for the nth 'diag_info_map.diag_name' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_DIAG_INFO_MAP<n>_DIAG_SOURCE
Specify the value for the nth 'diag_info_map.diag_source' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_DIAG_INFO_MAP<n>_FIELD_SOURCE
Specify the value for the nth 'diag_info_map.field_source' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_DIAG_INFO_MAP<n>_MATCH_TO_TRACK
Specify the value for the nth 'diag_info_map.match_to_track' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_DIAG_INFO_MAP<n>_TRACK_SOURCE
Specify the value for the nth 'diag_info_map.track_source' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_DIAG_SOURCE<n>
Specify the (optional) source string for the nth -diag argument for TCPairs.

Used by: TCPairs

TC_PAIRS_DIAG_TEMPLATE<n>
Specify the (optional) template for the nth -diag argument for TCPairs.

Used by: TCPairs

TC_PAIRS_DIR

> **Warning: DEPRECATED:** Please use *TC_PAIRS_OUTPUT_DIR*.

TC_PAIRS_DLAND_FILE
The file generated by the MET tool tc_dland, containing the gridded representation of the minimum distance to land. Please refer to the MET User's Guide for more information about the tc_dland tool.

Used by: TCPairs

TC_PAIRS_EDECK_INPUT_DIR
Directory that contains the EDECK files.

Used by: TCPairs

TC_PAIRS_EDECK_INPUT_TEMPLATE
Template of the file names of EDECK data.

Used by: TCPairs

TC_PAIRS_EDECK_TEMPLATE

> **Warning: DEPRECATED:** Please use *TC_PAIRS_EDECK_INPUT_TEMPLATE*.

TC_PAIRS_FORCE_OVERWRITE

> **Warning: DEPRECATED:** Please use *TC_PAIRS_SKIP_IF_OUTPUT_EXISTS*.

TC_PAIRS_INIT_BEG
Set the initialization begin time for TCpairs.

Used by: TCPairs

TC_PAIRS_INIT_END
Set the initialization end time for TCpairs.

Used by: TCPairs

TC_PAIRS_INIT_EXCLUDE
Specify which, if any, forecast initializations to exclude from the analysis.

Used by: TCPairs

TC_PAIRS_INIT_INCLUDE
Specify which forecast initializations to include in the analysis.

Used by: TCPairs

TC_PAIRS_INTERP12
Specify the value for 'interp12' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_MATCH_POINTS
Specify the value for 'match_points' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: TC_PAIRS_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: TCPairs

TC_PAIRS_MISSING_VAL
Specify the missing value code.

Used by: TCPairs

TC_PAIRS_MISSING_VAL_TO_REPLACE
Specify the missing value code to replace.

Used by: TCPairs

TC_PAIRS_MODEL

Warning: DEPRECATED: Please use *MODEL* instead.

TC_PAIRS_OUTPUT_DIR
Specify the directory where the MET tc_pairs tool will write files.

Used by: TCPairs

TC_PAIRS_OUTPUT_TEMPLATE
Template of the output file names created by tc_pairs.

Used by: TCPairs

TC_PAIRS_READ_ALL_FILES
Specify whether to pass the value specified in *TC_PAIRS_ADECK_INPUT_DIR*, *TC_PAIRS_BDECK_INPUT_DIR* and *TC_PAIRS_EDECK_INPUT_DIR* to the MET tc_pairs utility or have the wrapper search for valid files in that directory based on the value of *TC_PAIRS_ADECK_TEMPLATE*, *TC_PAIRS_BDECK_TEMPLATE* and *TC_PAIRS_EDECK_TEMPLATE* and pass them individually to tc_pairs. Set to false or no to have the wrapper find valid files. This can speed up execution time of tc_pairs.Acceptable values: yes/no

Used by: TCPairs

TC_PAIRS_REFORMAT_DECK
Set to true or yes if using cyclone data that needs to be reformatted to match the ATCF (Automated Tropical Cyclone Forecasting) format. If set to true or yes, you will need to set *TC_PAIRS_REFORMAT_TYPE* to specify which type of reformatting to perform.

Used by: TCPairs

TC_PAIRS_REFORMAT_DIR
Specify the directory to write reformatted track data to be read by tc_pairs. Used only if *TC_PAIRS_REFORMAT_DECK* is true or yes.

Used by: TCPairs

TC_PAIRS_REFORMAT_TYPE
Specify which type of reformatting to perform on cyclone data. Currently only SBU extra tropical cyclone reformatting is available. Only used if *TC_PAIRS_REFORMAT_DECK* is true or yes.Acceptable values: SBU

Used by: TCPairs

TC_PAIRS_RUN_ONCE
If True, TCPairs will be run once using the INIT_BEG or VALID_BEG value (depending on the value of LOOP_BY). This is the default setting and preserves the original logic of the wrapper. If this variable is

set to False, then TCPairs will run once for each run time iteration. The preferred configuration settings to run TCPairs once for a range of init or valid times is to set INIT_BEG to INIT_END (if LOOP_BY = INIT) and define the range of init times to filter the data inside TCPairs with TC_PAIRS_INIT_BEG and TC_PAIRS_INIT_END. The same applies for the VALID variables if LOOP_BY = VALID.

Used by: TCPairs

TC_PAIRS_SKIP_IF_OUTPUT_EXISTS

Specify whether to overwrite the output from the MET tc_pairs tool or not. If set to true or yes and the output file already exists for a given run, tc_pairs will not be run.Acceptable values: yes/no

Used by: TCPairs

TC_PAIRS_SKIP_IF_REFORMAT_EXISTS

Specify whether to overwrite the reformatted cyclone data or not. If set to true or yes and the reformatted file already exists for a given run, the reformatting code will not be run. Used only when *TC_PAIRS_REFORMAT_DECK* is set to true or yes.Acceptable values: yes/no

Used by: TCPairs

TC_PAIRS_SKIP_LEAD_SEQ

If True and a forecast lead sequence is set in the configuration, do not loop over list of leads and process for each. This is used for feature relative use cases where TCPairs is run for each storm initialization time and SeriesAnalysis is configured to filter the data by forecast leads. Default value is False.

Used by: TCPairs

TC_PAIRS_STORM_ID

The identifier of the storm(s) of interest.

Used by: TCPairs

TC_PAIRS_STORM_NAME

The name(s) of the storm of interest.

Used by: TCPairs

TC_PAIRS_VALID_BEG
Set the valid begin time for TCPairs.

Used by: TCPairs

TC_PAIRS_VALID_END
Set the valid end time for TCpairs.

Used by: TCpairs

TC_PAIRS_VALID_EXCLUDE
Specify the value for 'valid_exc' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_VALID_INCLUDE
Specify the value for 'valid_inc' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_WRITE_VALID
Specify the value for 'write_valid' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_RMW_BASIN
Specify the value for 'basin' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_CONFIG_FILE
Path to configuration file read by tc_rmw. If unset, parm/met_config/TCRMWConfig_wrapped will be used.

Used by: TCRMW

TC_RMW_CYCLONE
Specify the value for 'cyclone' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_DECK_INPUT_DIR
Directory containing ADECK input data to TCRMW. This variable is optional because you can specify the full path to the input files using *TC_RMW_DECK_TEMPLATE*.

Used by: TCRMW

TC_RMW_DECK_TEMPLATE
Filename template of the ADECK input data used by TCRMW. See also *TC_RMW_DECK_INPUT_DIR*.

Used by: TCRMW

TC_RMW_DELTA_RANGE_KM
Specify the value for 'delta_range_km' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_DESC
Specify the value for 'desc' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_INIT_INCLUDE
Value to set for init_include in the MET configuration file. See the MET User's Guide section regarding Regrid-Data-Plane for more information.

Used by: TCRMW

TC_RMW_INPUT_DATATYPE
Specify the data type of the input directory for input files used with the MET TCRMW tool. Used to set the 'file_type' value of the data dictionary in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_INPUT_DIR
Directory containing input data to TCRMW. This variable is optional because you can specify the full path to the input files using *TC_RMW_INPUT_TEMPLATE*.

Used by: TCRMW

TC_RMW_INPUT_FILE_LIST
Specifies an explicit path to a file list file to pass into tc_rmw. If set, *TC_RMW_INPUT_TEMPLATE* and *TC_RMW_INPUT_DIR* are ignored.

Used by: TCRMW

TC_RMW_INPUT_TEMPLATE
Filename template of the input data used by TCRMW. See also *TC_RMW_INPUT_DIR*.

Used by: TCRMW

TC_RMW_MAX_RANGE_KM
Specify the value for 'max_range_km' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: TC_RMW_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: TCRMW

TC_RMW_N_AZIMUTH
Specify the value for 'n_azimuth' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_N_RANGE
Specify the value for 'n_range' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_OUTPUT_DIR
Directory to write output data from TCRMW. This variable is optional because you can specify the full path to the output file using *TC_RMW_OUTPUT_TEMPLATE*.

Used by: TCRMW

TC_RMW_OUTPUT_TEMPLATE
Filename template of write the output data generated by TCRMW. See also *TC_RMW_OUTPUT_DIR*.

Used by: TCRMW

TC_RMW_REGRID_CENSOR_THRESH
Specify the value for 'regrid.censor_thresh' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_REGRID_CENSOR_VAL
Specify the value for 'regrid.censor_val' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_REGRID_CONVERT
Specify the value for 'regrid.convert' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_REGRID_METHOD
Specify the value for 'regrid.method' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_REGRID_SHAPE
Specify the value for 'regrid.shape' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_REGRID_VLD_THRESH
Specify the value for 'regrid.vld_thresh' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_REGRID_WIDTH
Specify the value for 'regrid.width' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_SCALE
Specify the value for 'rmw_scale' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: TCRMW

TC_RMW_STORM_ID
Specify the value for 'storm_id' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_STORM_NAME
Specify the value for 'storm_name' in the MET configuration file for TCRMW.

Used by: TCRMW

TC_RMW_VALID_BEG
Value to set for valid_beg in the MET configuration file. See the MET User's Guide section regarding Regrid-Data-Plane for more information.

Used by: TCRMW

TC_RMW_VALID_END
Value to set for valid_end in the MET configuration file. See the MET User's Guide section regarding Regrid-Data-Plane for more information.

Used by: TCRMW

TC_RMW_VALID_EXCLUDE_LIST
List of values to set for valid_exc in the MET configuration file. See the MET User's Guide section regarding Regrid-Data-Plane for more information.

Used by: TCRMW

TC_RMW_VALID_HOUR_LIST
List of values to set for valid_hour in the MET configuration file. See the MET User's Guide section regarding Regrid-Data-Plane for more information.

Used by: TCRMW

TC_RMW_VALID_INCLUDE_LIST
List of values to set for valid_inc in the MET configuration file. See the MET User's Guide section regarding Regrid-Data-Plane for more information.

Used by: TCRMW

TC_STAT_AMODEL
Specify the AMODEL for the MET tc_stat tool.

Used by: TCStat

TC_STAT_BASIN
Specify the BASIN for the MET tc_stat tool.

Used by: TCStat

TC_STAT_BMODEL
Specify the BMODEL for the MET tc_stat tool.

Used by: TCStat

TC_STAT_CMD_LINE_JOB

> **Warning: DEPRECATED:** Please set *TC_STAT_CONFIG_FILE* to run using a config file and leave it unset to run via the command line.

Old: Specify expression(s) that will be passed to the MET tc_stat tool via the command line. Only specify if TC_STAT_RUN_VIA=CLI. Please refer to the MET User's Guide chapter for tc-stat for the details on performing job summaries and job filters.

Used by: TCStat

TC_STAT_COLUMN_STR_EXC_NAME
Specify the value for 'column_str_exc_name' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_COLUMN_STR_EXC_VAL
Specify the value for 'column_str_exc_val' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_COLUMN_STR_NAME
Specify the string names of the columns for stratification with the MET tc_stat tool.

Used by: TCStat

TC_STAT_COLUMN_STR_VAL
Specify the values for the columns set via the *TC_STAT_COLUMN_STR_NAME* option for use with the MET tc_stat tool.

Used by: TCStat

TC_STAT_COLUMN_THRESH_NAME
Specify the string names of the columns for stratification by threshold with the MET tc_stat tool.

Used by: TCStat

TC_STAT_COLUMN_THRESH_VAL
Specify the values used for thresholding the columns specified in the *TC_STAT_COLUMN_THRESH_NAME* option for use with the MET tc_stat tool.

Used by: TCStat

TC_STAT_CONFIG_FILE
Path to configuration file read by tc_stat. If unset, parm/met_config/TCStatConfig_wrapped will be used.

Used by: TCStat

TC_STAT_CYCLONE
Specify the cyclone of interest for use with the MET tc_stat tool.

Used by: TCStat

TC_STAT_DESC
Specify the desc option for use with the MET tc_stat tool.

Used by: TCStat

TC_STAT_DIAG_THRESH_NAME
Specify the value for 'diag_thresh_name' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_DIAG_THRESH_VAL
Specify the value for 'diag_thresh_val' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_EVENT_EQUAL
Specify the value for 'event_equal' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_EVENT_EQUAL_LEAD
Specify the value for 'event_equal_lead' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_INIT_BEG
Specify the beginning initialization time for stratification when using the MET tc_stat tool. Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCStat

TC_STAT_INIT_DIAG_THRESH_NAME
Specify the value for 'init_diag_thresh_name' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_INIT_DIAG_THRESH_VAL
Specify the value for 'init_diag_thresh_val' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_INIT_END
Specify the ending initialization time for stratification when using the MET tc_stat tool. Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCStat

TC_STAT_INIT_EXCLUDE
Specify the initialization times to exclude when using the MET tc_stat tool, via a comma separated list e.g.:20141220_18, 20141221_00Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCStat

TC_STAT_INIT_HOUR
The beginning hour (HH) of the initialization time of interest.

Used by: TCStat

TC_STAT_INIT_INCLUDE
Specify the initialization times to include when using the MET tc_stat tool, via a comma separated list e.g.:20141220_00, 20141220_06, 20141220_12Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCStat

TC_STAT_INIT_MASK
This corresponds to the INIT_MASK keyword in the MET tc_stat config file. For more information, please refer to the MET User's Guide .

Used by: TCStat

TC_STAT_INIT_STR_EXC_NAME
Specify the value for 'init_str_exc_name' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_INIT_STR_EXC_VAL
Specify the value for 'init_str_exc_val' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_INIT_STR_NAME
This corresponds to the INIT_STR_NAME keyword in the MET tc_stat config file. Please refer to the MET User's Guide for more details.

Used by: TCStat

TC_STAT_INIT_STR_VAL
This corresponds to the INIT_STR_VAL keyword in the MET tc_stat config file. Please refer to the MET User's Guide for more information.

Used by: TCStat

TC_STAT_INIT_THRESH_NAME
Specify the string names of the columns for stratification by threshold with the MET tc_stat tool.

Used by: TCStat

TC_STAT_INIT_THRESH_VAL
Specify the values used for thresholding the columns specified in the *TC_STAT_INIT_THRESH_NAME* option for use with the MET tc_stat tool.

Used by: TCStat

TC_STAT_INPUT_DIR

> **Warning:** **DEPRECATED:** Please use *TC_STAT_LOOKIN_DIR*.

Used by: TCStat

TC_STAT_JOB_ARGS
Specify expressions for the MET tc_stat tool to execute.

Used by: TCStat

TC_STAT_JOBS_LIST

> **Warning: DEPRECATED:** Please use *TC_STAT_JOB_ARGS*.

TC_STAT_LANDFALL
Specify whether only those points occurring near landfall should be retained when using the MET tc_stat tool. Acceptable values: True/False

Used by: TCStat

TC_STAT_LANDFALL_BEG
Specify the beginning of the landfall window for use with the MET tc_stat tool. Acceptable formats: HH, HHmmss

Used by: TCStat

TC_STAT_LANDFALL_END
Specify the end of the landfall window for use with the MET tc_stat tool. Acceptable formats: HH, HHmmss

Used by: TCStat

TC_STAT_LEAD
Specify the lead times to stratify by when using the MET tc_stat tool. Acceptable formats: HH, HHmmss

Used by: TCStat

TC_STAT_LEAD_REQ
Specify the LEAD_REQ when using the MET tc_stat tool.

Used by: TCStat

TC_STAT_LINE_TYPE
Specify the value for 'line_type' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_LOOKIN_DIR
Specify the input directory where the MET tc_stat tool will look for files.

Used by: TCStat

TC_STAT_MATCH_POINTS
Specify whether only those points common to both the ADECK and BDECK tracks should be written out or not when using the MET tc_stat tool. Acceptable values: True/False

Used by: TCStat

TC_STAT_MET_CONFIG_OVERRIDES
Override any variables in the MET configuration file that are not supported by the wrapper. This should be set to the full variable name and value that you want to override, including the equal sign and the ending semi-colon. The value is directly appended to the end of the wrapped MET config file.

Example: TC_STAT_MET_CONFIG_OVERRIDES = desc = "override_desc"; model = "override_model";

See *Overriding Unsupported MET config file settings* (page 87) for more information

Used by: TCStat

TC_STAT_OUT_INIT_MASK
Specify the value for 'out_init_mask' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_OUT_VALID_MASK
Specify the value for 'out_valid_mask' in the MET configuration file for TCStat.

Used by: TCStat

TC_STAT_OUTPUT_DIR
Specify the output directory where the MET tc_stat tool will write files.

Used by: TCStat

TC_STAT_OUTPUT_TEMPLATE
(Optional) Specify the template of the output file to write job output from tc_stat. If set, then the -out command line argument with the full path to the file will be added to the tc_stat call.

Used by: TCStat

TC_STAT_RUN_VIA

> **Warning: DEPRECATED:** Please set *TC_STAT_CONFIG_FILE* to run using a config file and leave it unset to run via the command line.

Old: Specify the method for running the MET tc_stat tool. Acceptable values: CONFIG. If left blank (unset), tc_stat will run via the command line.

Used by: TCStat

TC_STAT_SKIP_IF_OUTPUT_EXISTS
If True, do not run app if output file already exists. Set to False to overwrite files.

Used by: TCStat

TC_STAT_STORM_ID
Set the STORM_ID(s) of interest with the MET tc_stat tool.

Used by: TCStat

TC_STAT_STORM_NAME
Set the environment variable STORM_NAME for use with the MET tc_stat tool.

Used by: TCStat

TC_STAT_TRACK_WATCH_WARN
Specify which watches and warnings to stratify over when using the MET tc_stat tool. Acceptable values: HUWARN, HUWATCH, TSWARN, TSWATCH, ALLIf left blank (unset), no stratification will be done.

Used by: TCStat

TC_STAT_VALID_BEG
Specify a comma separated list of beginning valid times to stratify with when using the MET tc_stat tool. Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCStat

TC_STAT_VALID_END
Specify a comma separated list of ending valid times to stratify with when using the MET tc_stat tool. Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCStat

TC_STAT_VALID_EXCLUDE
Specify a comma separated list of valid times to exclude from the stratification with when using the MET tc_stat tool. Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCStat

TC_STAT_VALID_HOUR
This corresponds to the VALID_HOUR keyword in the MET tc_stat config file. For more information, please refer to the MET User's Guide.

Used by: TCStat

TC_STAT_VALID_INCLUDE
Specify a comma separated list of valid times to include in the stratification with when using the MET tc_stat tool. Acceptable formats: YYYYMMDD_HH, YYYYMMDD_HHmmss

Used by: TCStat

TC_STAT_VALID_MASK
This corresponds to the VALID_MASK in the MET tc_stat config file. Please refer to the MET User's Guide for more information.

Used by: TCStat

TC_STAT_WATER_ONLY
Specify whether to exclude points where the distance to land is <= 0. If set to TRUE, once land is encountered the remainder of the forecast track is not used for the verification, even if the track moves back over water.Acceptable values: true/false

Used by: TCStat

TCMPR_DATA_DIR

Warning: DEPRECATED: Please use *TCMPR_PLOTTER_TCMPR_DATA_DIR*.

TCMPR_PLOT_OUT_DIR

Warning: DEPRECATED: Please use *TCMPR_PLOTTER_PLOT_OUTPUT_DIR*.

TCMPR_PLOTTER_CONFIG_FILE
Configuration file used by TCMPRPlotter.

Used by: TCMPRPlotter

TCMPR_PLOTTER_DEMO_YR
The demo year. This is an optional value used by the plot_TCMPR.R script, (which is wrapped by TCMPRPlotter). Please refer to the MET User's Guide for more details.

Used by: TCMPRPlotter

TCMPR_PLOTTER_DEP_LABELS
List of strings that correspond to the values in *TCMPR_PLOTTER_DEP_VARS* that can be referenced in

other variables to set the plot title, axis labels, etc. with the {dep_label} tag.

Used by: TCMPRPlotter

TCMPR_PLOTTER_DEP_VARS

Corresponds to the optional flag -dep in the plot_TCMPR.R script, which is wrapped by TCMPRPlotter. The value to this flag is a comma-separated list (no whitespace) of dependent variable columns to plot (e.g. AMSLP-BMSLP, AMAX_WIND-BMAX_WIND, TK_ERR). If this is undefined, then the default plot for TK_ERR (track error) is generated. The values in this list are looped over to run once for each and can be referenced in other variables using the {dep} tag. Note, if you want the track error plot generated, in addition to other plots, then you need to explicitly list this with the other variables. Please refer to the MET User's Guide for more details.

Used by: TCMPRPlotter

TCMPR_PLOTTER_FILTER

Corresponds to the optional -filter argument to the plot_TCMPR.R script which is wrapped by TCMPRPlotter. This is a list of filtering options for the tc_stat tool.

Used by: TCMPRPlotter

TCMPR_PLOTTER_FILTERED_TCST_DATA_FILE

Corresponds to the optional -tcst argument to the plot_TCMPR.R script which is wrapped by TCMPRPlotter. This is a tcst data file to be used instead of running the tc_stat tool. Indicate a full path to the data file.

Used by: TCMPRPlotter

TCMPR_PLOTTER_FOOTNOTE_FLAG

This corresponds to the optional -footnote flag in the plot_TCMPR.R script which is wrapped by TCMPRPlotter. According to the plot_TCMPR.R usage, this flag is used to disable footnote (date).

Used by: TCMPRPlotter

TCMPR_PLOTTER_HFIP_BASELINE

Corresponds to the optional -hfip_bsln flag in the plot_TCMPR.R script which is wrapped by TCMPRPlotter. This is a string that indicates whether to add the HFIP baseline, and indicates the version (no, 0, 5, 10 year goal).

Used by: TCMPRPlotter

TCMPR_PLOTTER_LEAD
For CyclonePlotter, this refers to the column of interest in the input ASCII cyclone file.In the TCMPRPlotter, this corresponds to the optional -lead argument in the plot_TCMPR.R script (which is wrapped by TCMPRPlotter). This argument is set to a comma-separted list of lead times (h) to be plotted.In TCStat, this corresponds to the name of the column of interest in the input ASCII data file.

Used by: TCMPRPlotter

TCMPR_PLOTTER_LEGEND
The text to be included in the legend of your plot.

Used by: TCMPRPlotter

TCMPR_PLOTTER_NO_EE
Set the *NO_EE* flag for the TC Matched Pairs plotting utility.Acceptable values: yes/no

Used by: TCMPRPlotter

TCMPR_PLOTTER_NO_LOG
Set the NO_LOG flag for the TC Matched Pairs plotting utility.Acceptable values: yes/no

Used by: TCMPRPlotter

TCMPR_PLOTTER_PLOT_CONFIG_OPTS
Specify plot configuration options for the TC Matched Pairs plotting tool.

Used by: TCMPRPlotter

TCMPR_PLOTTER_PLOT_LABELS
List of strings that correspond to the values in *TCMPR_PLOTTER_PLOT_TYPES* that can be referenced in other variables to set the plot title, axis labels, etc. with the {plot_label} tag.

Used by: TCMPRPlotter

TCMPR_PLOTTER_PLOT_OUTPUT_DIR
Provide the output directory where the TC Matched Pairs plotting tool will create files.

Used by: TCMPRPlotter

TCMPR_PLOTTER_PLOT_TYPES
Specify what plot types are desired for the TC Matched Pairs plotting tool. By default, a boxplot is generated if this is undefined in the configuration file. If other plots are requested and a boxplot is also desired, you must explicitly list boxplot in your list of plot types. Supported plot types: BOXPLOT, POINT, MEAN, MEDIAN, RELPERF (relative performance), RANK (time series of ranks for the first model), SCATTER, SKILL_MN (mean skill scores) and SKILL_MD (median skill scores). The values in this list are looped over to run once for each and can be referenced in other variables using the {plot} tag.

Used by: TCMPRPlotter

TCMPR_PLOTTER_PREFIX
Prefix used in TCMPRPlotter.

Used by: TCMPRPlotter

TCMPR_PLOTTER_READ_ALL_FILES
If True, pass in input directory set by *TCMPR_PLOTTER_TCMPR_DATA_DIR* to the script. If False, a list of all files that end with .tcst in the input directory is gathered and passed into the script. Defaults to False.

Used by: TCMPRPlotter

TCMPR_PLOTTER_RP_DIFF
This corresponds to the optional -rp_diff flag of the plot_TCMPR.R script (which is wrapped by TCMPRPlotter). This a comma-separated list of thresholds to specify meaningful differences for the relative performance plot.

Used by: TCMPRPlotter

TCMPR_PLOTTER_SAVE
Corresponds to the optional -save flag in plot_TCMPR.R (which is wrapped by TCMPRPlotter). This is a yes/no value to indicate whether to save the image (yes).

Used by: TCMPRPlotter

TCMPR_PLOTTER_SAVE_DATA
Corresponds to the optional -save_data flag in plot_TCMPR.R (which is wrapped by TCMPRPlotter). Indicates whether to save the filtered track data to a file instead of deleting it.

Used by: TCMPRPlotter

TCMPR_PLOTTER_SCATTER_X
Corresponds to the optional -scatter_x flag in plot_TCMPR.R (which is wrapped by TCMPRPlotter). This is a comma-separated list of x-axis variable columns to plot.

Used by: TCMPRPlotter

TCMPR_PLOTTER_SCATTER_Y
Corresponds to the optional -scatter_y flag in plot_TCMPR.R (which is wrapped by TCMPRPlotter). This is a comma-separated list of y-axis variable columns to plot.

Used by: TCMPRPlotter

TCMPR_PLOTTER_SERIES
Corresponds to the optional -series flag in plot_TCMPR.R (which is wrapped by TCMPRPlotter). This is the column whose unique values define the series on the plot, optionally followed by a comma-separated list of values, including: ALL, OTHER, and colon-separated groups.

Used by: TCMPRPlotter

TCMPR_PLOTTER_SERIES_CI
Corresponds to the optional -series_ci flag in plot_TCMPR.R (which is wrapped by TCMPRPlotter). This is a list of true/false for confidence intervals. This list can be optionally followed by a comma-separated list of values, including ALL, OTHER, and colon-separated groups.

Used by: TCMPRPlotter

TCMPR_PLOTTER_SKILL_REF
This corresponds to the optional -skill_ref flag in plot_TCMPR.R (which is wrapped by TCMPRPlotter). This is the identifier for the skill score reference.

Used by: TCMPRPlotter

TCMPR_PLOTTER_SUBTITLE
The subtitle of the plot.

Used by: TCMPRPlotter

TCMPR_PLOTTER_TCMPR_DATA_DIR
Provide the input directory for the track data for the TC Matched Pairs plotting tool.

Used by: TCMPRPlotter

TCMPR_PLOTTER_TITLE
Specify a title string for the TC Matched Pairs plotting tool.

Used by: TCMPRPlotter

TCMPR_PLOTTER_XLAB
Specify the x-axis label when using the TC Matched Pairs plotting tool.

Used by: TCMPRPlotter

TCMPR_PLOTTER_XLIM
Specify the x-axis limit when using the TC Matched Pairs plotting tool.

Used by: TCMPRPlotter

TCMPR_PLOTTER_YLAB
Specify the y-axis label when using the TC Matched Pairs plotting tool.

Used by: TCMPRPlotter

TCMPR_PLOTTER_YLIM
Specify the y-axis limit when using the TC Matched Pairs plotting tool.

Used by: TCMPRPlotter

TIME_METHOD

> **Warning:** **DEPRECATED:** Please use *LOOP_BY* instead.

TIME_SUMMARY_BEG

> **Warning:** **DEPRECATED:** Please use *PB2NC_TIME_SUMMARY_BEG* instead.

TIME_SUMMARY_END

> **Warning:** **DEPRECATED:** Please use *PB2NC_TIME_SUMMARY_END* instead.

TIME_SUMMARY_FLAG

> **Warning:** **DEPRECATED:** Please use *PB2NC_TIME_SUMMARY_FLAG* instead.

TIME_SUMMARY_TYPES

> **Warning:** **DEPRECATED:** Please use *PB2NC_TIME_SUMMARY_TYPES* instead.

TIME_SUMMARY_VAR_NAMES

> **Warning:** **DEPRECATED:** Please use *PB2NC_TIME_SUMMARY_VAR_NAMES* instead.

TITLE

> **Warning:** **DEPRECATED:** Please use *TCMPR_PLOTTER_TITLE* instead.

TMP_DIR
Specify the path to a temporary directory where the user has write permissions.

Used by: PB2NC, PointStat, TCStat

TOP_LEVEL_DIRS

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_READ_ALL_FILES*.

TR

Specify the path to the Linux "tr" executable.

Used by: PB2NC, PointStat

TR_EXE

> **Warning:** **DEPRECATED:** Please use *TR*.

TRACK_DATA_DIR

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_ADECK_INPUT_DIR*, *TC_PAIRS_BDECK_INPUT_DIR* and *TC_PAIRS_EDECK_INPUT_DIR*.

TRACK_DATA_MOD_FORCE_OVERWRITE

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_SKIP_IF_REFORMAT_EXISTS*.

TRACK_DATA_SUBDIR_MOD

> **Warning:** **DEPRECATED:** No longer used.

TRACK_TYPE

> **Warning:** **DEPRECATED:** Please use *TC_PAIRS_REFORMAT_DECK*.

USER_SCRIPT_COMMAND

User-defined command to run. Filename template tags can be used to modify the command for each execution. See *USER_SCRIPT_RUNTIME_FREQ* for more information.

Used by: UserScript

USER_SCRIPT_CUSTOM_LOOP_LIST
List of strings to loop over for each runtime to run the command.

Used by: UserScript

USER_SCRIPT_INPUT_DIR
Optional directory to look for input files. Prepended to each input template (see *USER_SCRIPT_INPUT_TEMPLATE*).

Used by: UserScript

USER_SCRIPT_INPUT_TEMPLATE
Optional list of input templates to use to look for input files. If *USER_SCRIPT_INPUT_DIR* is set, prepend that path to each item. When the UserScript wrapper is run, the templates defined here will be used to populate a list of all of the files that match the template for each run time specified. Depending on the runtime frequency defined in *USER_SCRIPT_RUNTIME_FREQ*, text files will be generated that contain a list of the file paths that correspond to the current run. If any files are not found on disk, then "missing" will be added in place of the file path. Each file list text file will be named after the current init/valid/lead values for that run and a label named input<n> where <n> is a zero-based index of the template, i.e. a single template will be labelled input0, two templates will be labelled input0 and input1, etc. Custom labels can be defined with *USER_SCRIPT_INPUT_TEMPLATE_LABELS*. For each template, an environment variable named METPLUS_FILELIST_<label> will be set to the path of the appropriate file list text file. This environment variable can be referenced by the user-defined script to obtain the file list.

Used by: UserScript

USER_SCRIPT_INPUT_TEMPLATE_LABELS
Optional list of labels that correspond to each input template defined. See *USER_SCRIPT_INPUT_TEMPLATE*. Each template that does not have a label defined will be assigned a label with the format input<n> where <n> is the zero-based index of the template in the list.

Used by: UserScript

USER_SCRIPT_RUNTIME_FREQ
Frequency to run the user-defined script. See *Runtime Frequency* (page 70) for more information.

Used by: UserScript

USER_SCRIPT_SKIP_TIMES
Run times to skip for this wrapper only. See *SKIP_TIMES* for more information and how to format.

Used by: UserScript

VALID_BEG
Specify a begin time for valid times for use in the analysis. This is the starting date in the format set in the *VALID_TIME_FMT*. It is named accordingly to the value set for *LOOP_BY*. However, in StatAnalysis, it is named accordingly to the value set for *PLOT_TIME*. See *Looping by Valid Time* (page 47) for more information.

Used by: All

VALID_END
Specify an end time for valid times for use in the analysis. This is the ending date in the format set in the *VALID_TIME_FMT*. It is named accordingly to the value set for *LOOP_BY*. See *Looping by Valid Time* (page 47) for more information.

Used by: All

VALID_HOUR_BEG

Warning: **DEPRECATED:** Please use *FCST_VALID_HOUR_LIST* or *OBS_VALID_HOUR_LIST* instead.

VALID_HOUR_END

Warning: **DEPRECATED:** Please use *FCST_VALID_HOUR_LIST* or *OBS_VALID_HOUR_LIST* instead.

VALID_HOUR_INCREMENT

Warning: **DEPRECATED:** Please use *FCST_VALID_HOUR_LIST* or *OBS_VALID_HOUR_LIST* instead.

VALID_HOUR_METHOD

Warning: **DEPRECATED:** No longer used.

VALID_INCREMENT

Specify the time increment for valid times for use in the analysis. See *Looping by Valid Time* (page 47) for more information. Units are assumed to be seconds unless specified with Y, m, d, H, M, or S.

Used by: All

VALID_LIST

List of valid times to process. This variable is used when intervals between run times are irregular. It is only read if *LOOP_BY* = VALID. If it is set, then *VALID_BEG*, *VALID_END*, and *VALID_INCREMENT* are ignored. All values in the list must match the format of *VALID_TIME_FMT* or they will be skipped.

Used by: All

VALID_TIME_FMT

Specify a strftime formatting string for use with *VALID_BEG* and *VALID_END*. See *Looping by Valid Time* (page 47) for more information.

Used by: All

VAR<n>_FOURIER_DECOMP

Specify if Fourier decomposition is to be considered (True) or not (False). If this is set to True, data stratification will be done for the Fourier decomposition of FCS_VAR<n>_NAME. This should have been previously run in grid_stat_wrapper. The default value is set to False.

Used by: StatAnalysis

VAR<n>_WAVE_NUM_LIST

Specify a comma separated list of wave numbers pairings of the Fourier decomposition.

Used by: StatAnalysis

VAR_LIST

> **Warning:** **DEPRECATED:** Please use *FCST_VAR<n>_NAME* and *OBS_VAR<n>_NAME* instead.

VERIFICATION_GRID

Warning: **DEPRECATED:** Please use *REGRID_DATA_PLANE_VERIF_GRID* instead.

VERTICAL_LOCATION

Warning: **DEPRECATED:** Specify the vertical location desired when using the MET pb2nc tool.

VX_MASK_LIST

Specify the values of the VX_MASK column in the MET .stat file to use; a list of the verification regions of interest.

Groups of values can be looped over by setting VX_MASK_LIST<n> and adding VX_MASK_LIST to *LOOP_LIST_ITEMS*. See *Looping Over Groups of Lists* (page 240) for more information.

Used by: StatAnalysis

XLAB

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_XLAB* instead.

XLIM

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_XLIM* instead.

YLAB

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_YLAB* instead.

YLIM

Warning: **DEPRECATED:** Please use *TCMPR_PLOTTER_YLIM* instead.

Chapter 10

METplus Statistics & Diagnostics

10.1 Statistics Database

10.1.1 Statistics List A-B

Table 1: Statistics List A-B

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Accuracy	ACC	Categorical	Point-Stat Grid-Stat MODE	CTS MCTS NBRCTS MODE cts
Asymptotic Fractions Skill Score	AFSS	Neighborhood	Grid-Stat	NBRCNT
Along track error (nm)	ALTK_ERR	Continuous	TC-Pairs TC-Stat	TCMPR TCST
Anomaly Correlation including mean error	ANOM_CORR	Continuous	Point-Stat Grid-Stat Series-Analysis Stat-Analysis	CNT
Uncentered Anomaly Correlation excluding mean error	ANOM_CORR_UNCNTR	Continuous	Point-Stat Grid-Stat Series-Analysis Stat-Analysis	CNT
Baddeley's Delta Metric	BADDELEY	Distance Map	Grid-Stat	DMAP
Bias Adjusted Gilbert Skill Score	BAGSS	Categorical	Point-Stat Grid-Stat	CTS NBRCTS
Base Rate	BASER	Categorical	Point-Stat Grid-Stat Wavelet-Stat MODE	CTS ECLV MODE cts NBRCTCS PSTD PJC
Bias-corrected mean squared error	BCMSE	Continuous	Point-Stat Grid-Stat Ensemble-Stat	CNT SSVAR
Brier Score	BRIER	Probability	Point-Stat Grid-Stat	PSTD
Climatological Brier Score	BRIERCL	Probability	Point-Stat Grid-Stat	PSTD
Brier Skill Score relative to	BSS	Proba-	Point-Stat Grid-Stat	PSTD

10.1.2 Statistics List C-E

Table 2: Statistics List C-E

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Calibration when forecast is between the ith and i+1th probability thresholds (repeated)	CALIBRATION _i	Probability	Point-Stat Grid-Stat	PJC
Climatological mean value	CLIMO_MEAN	Continuous	Point-Stat Grid-Stat Ensemble-Stat	MPR ORANK
Climatological standard deviation value	CLIMO_STDEV	Continuous	Point-Stat Grid-Stat Ensemble-Stat	MPR ORANK
Continuous Ranked Probability Score (normal dist.)	CRPS	Ensemble	Ensemble-Stat	ECNT
Continuous Ranked Probability Score (empirical dist.)	CRPS_EMP	Ensemble	Ensemble-Stat	ECNT
Climatological Continuous Ranked Probability Score (normal dist.)	CRPSCL	Ensemble	Ensemble-Stat	ECNT
Climatological Continuous Ranked Probability Score (empirical dist.)	CRPSCL_EMP	Ensemble	Ensemble-Stat	ECNT
Continuous Ranked Probability Skill Score (normal dist.)	CRPSS	Ensemble	Ensemble-Stat	ECNT
Continuous Ranked Probability Skill Score (empirical dist.)	CRPSS_EMP	Ensemble	Ensemble-Stat	ECNT
Cross track error (nm)	CRTK_ERR	Continuous	TC-Pairs TC-Stat	TCMPR TCST
Critical Success Index	CSI	Categorical	Point-Stat MODE cts Grid-Stat	CTS MODE MBRCTCS
Absolute value of DIR_ERR (see below)	DIR_ABSERR	Continuous	Point-Stat Grid-Stat	VCNT
Signed angle between the directions of the average forecast and observed wind vectors	DIR_ERR	Continuous	Point-Stat Grid-Stat	VCNT
Expected correct rate used for MCTS HSS_EC	EC_VALUE	Categorical	Point-Stat Grid-Stat	MCTC
Extreme Dependency Index	EDI	Categorical	Point-Stat Grid-Stat	CTS NBRCTS
Extreme Dependency Score	EDS	Categorical	Point-Stat Grid-Stat	CTS NBRCTS
Mean of absolute value of forecast minus observed gradients	EGBAR	Continuous	Grid-Stat	GRAD
The unperturbed ensemble mean value	ENS_MEAN	Ensemble	Ensemble-Stat	ORANK
The PERTURBED ensemble mean (e.g. with Observation Error).	ENS_MEAN_OERR	Ensemble	Ensemble-Stat	ORANK
Standard deviation of the error	ESTDEV	Continuous	Point-Stat Grid-Stat Ensemble-	CNT SSVAR

10.1.3 Statistics List F

Table 3: Statistics List F

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Forecast rate/event frequency	F_RATE	Categorical	Point-Stat Grid-Stat	FHO NBRCNT
Mean forecast wind speed	F_SPEED_BAR	Continuous	Point-Stat Grid-Stat	VL1L2
Mean Forecast Anomaly	FABAR	Continuous	Point-Stat Grid-Stat	SAL1L2
False alarm ratio	FAR	Categorical	Point-Stat Grid-Stat MODE	CTS MODE NBRCTCS
Forecast mean	FBAR	Categorical	Ensemble-Stat Point-Stat Grid-Stat	SSVAR CNT SL1L2 VCNT
Length (speed) of the average forecast wind vector	FBAR_SPEED	Continuous	Point-Stat Grid-Stat	VCNT
Frequency Bias	FBIAS	Categorical	Wavelet-Stat MODE Point-Stat Grid-Stat	ISC MODE CTS NBRCTCS DMAP
Fractions Brier Score	FBS	Continuous	Grid-Stat	NBRCNT
Direction of the average forecast wind vector	FDIR	Continuous	Point-Stat Grid-Stat	VCNT
Mean Forecast Anomaly Squared	FFABAR	Continuous	Point-Stat Grid-Stat	SAL1L2
Average of forecast squared.	FFBAR	Continuous	Ensemble-Stat Point-Stat Grid-Stat	SSVAR SL1L2
Count of events in forecast category i and observation category j	Fi_Oj	Categorical	Point-Stat Grid-Stat	MCTC
Forecast mean	FMEAN	Continuous	MODE Grid-Stat Point-Stat	MODE NBRCTCS CTS
Number of forecast no and observation no	FN_ON	Categorical	MODE Grid-Stat Point-Stat	MODE NBRCTC CTC
Number of forecast no and observation yes	FN_OY	Categorical	MODE Grid-Stat Point-Stat	MODE NBRCTC CTC
Attributes for pairs of simple forecast and observation objects	FNNN_ONNN	Categorical	MODE	MODE obj
Average product of forecast-climo and observation-climo / Mean(f-c)*(o-c)	FOABAR	Continuous	Point-Stat Grid-Stat	SAL1L2
Average product of forecast and observation / Mean(f*o)	FOBAR	Continuous	Ensemble-Stat Point-Stat Grid-Stat	SSVAR SL1L2
Number of tied forecast ranks used in computing Kendall's tau statistic	FRANK_TIES	Continuous	Point-Stat Grid-Stat	CNT
Root mean square forecast wind speed	FS_RMS	Continuous	Point-Stat Grid-Stat	VCNT
Fractions Skill Score	FSS	Neighborhood	Grid-Stat	NBRCNT
Standard deviation of the error	FSTDEV	Continuous	Ensemble-Stat Point-Stat Grid-Stat	SSVAR CNT VCNT
Number of forecast events	FY	Categorical	Grid-Stat	DMAP

10.1.4 Statistics List G-M

Table 4: Statistics List G-M

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Gerrity Score and bootstrap confidence limits	GER	Categorical	Point-Stat Grid-Stat	MCTS
Gilbert Skill Score	GSS	Categorical	Point-Stat Grid-Stat MODE	CTS NBRCTCS MODE
Hit rate	H_RATE	Categorical	Point-Stat Grid-Stat	FHO
Hanssen and Kuipers Discriminant	HK	Categorical	MODE Point-Stat Grid-Stat	MODE cts MCTS CTS NBRCTS
Heidke Skill Score	HSS	Categorical	MODE Point-Stat Grid-Stat	MODE cts MCTS CTS NBRCTS
Heidke Skill Score user-specific expected correct	HSS_EC	Categorical	Point-Stat Grid-Stat	MCTS
Ignorance Score	IGN	Ensemble	Ensemble-Stat	ECNT
Interquartile Range	IQR	Continuous	Point-Stat Grid-Stat	CNT
Kendall's tau statistic	KT_CORR	Continuous	Point-Stat Grid-Stat	CNT
Likelihood when forecast is between the ith and i+1th probability thresholds repeated	LIKELIHOOD_i	Probability	Point-Stat Grid-Stat	PJC
Logarithm of the Odds Ratio	LODDS	Categorical	Point-Stat Grid-Stat	CTS NBRCTS
The Median Absolute Deviation	MAD	Continuous	Point-Stat Grid-Stat	CNT
Mean absolute error	MAE	Continuous	Point-Stat Grid-Stat	CNT SAL1L2 SL1L2
Magnitude & Multiplicative bias	MBIAS	Continuous	Ensemble-Stat Point-Stat Grid-Stat	SSVAR CNT
The Mean Error	ME	Continuous	Ensemble-Stat Point-Stat Grid-Stat	ECNT SSVAR CNT
The Mean Error of the PERTURBED en-	ME_OERR	Con-	Ensemble-Stat	ECNT

10.1.5 Statistics List N-O

Table 5: Statistics List N-O

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Dimension of the contingency table & the total number of categories in each dimension	N_CAT	Categorical	Point-Stat Grid-Stat	MCTC MCTS
Observation rate	O_RATE	Categorical	Point-Stat Grid-Stat	NBRCNT FHO
Mean observed wind speed	O_SPEED_BAR	Continuous	Point-Stat Grid-Stat	VL1L2
Mean Observation Anomaly	OABAR	Continuous	Point-Stat Grid-Stat	SAL1L2
Average observed value	OBAR	Continuous	Ensemble-Stat Point-Stat Grid-Stat .	SSVAR CNT SL1L2 VCNT
Length (speed) of the average observed wind vector	OBAR_SPEED	Continuous	Point-Stat Grid-Stat	VCNT
Odds Ratio	ODDS	Categorical	MODE Point-Stat Grid-Stat	MODE CTS NBRCTS
Direction of the average observed wind vector	ODIR	Continuous	Point-Stat Grid-Stat	VCNT
Number of observation when forecast is between the ith and i+1th probability thresholds	ON_i	Probability	Point-Stat Grid-Stat	PTC
Number of observation when forecast is between the ith and i+1th probability thresholds	ON_TP_i	Probability	Point-Stat Grid-Stat	PJC
Mean Squared Observation Anomaly	OOABAR	Continuous	Point-Stat Grid-Stat	SAL1L2
Average of observation squared	OOBAR	Continuous	Ensemble-Stat Point-Stat Grid-Stat	SSVAR SL1L2
Number of tied observation ranks used in computing Kendall's tau statistic	ORANK_TIES	Continuous	Point-Stat Grid-Stat	CNT
Odds Ratio Skill Score	ORSS	Categorical	Point-Stat Grid-Stat	CTS NBRCTS
Root mean square observed wind speed	OS_RMS	Continuous	Point-Stat Grid-Stat	VCNT

10.1.6 Statistics List P-R

Table 6: Statistics List P-R

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Probability Integral Transform	PIT	Ensemble	Ensemble-Stat	ORANK
Probability of false detection	PODF	Categorical	Point-Stat Grid-Stat	CTS
Probability of detecting no	PODN	Categorical	Point-Stat Grid-Stat MODE	CTS NBRCTCS MODE
Probability of detecting yes	PODY	Categorical	Point-Stat Grid-Stat MODE	CTS NBRCTCS MODE
Probability of detecting yes when forecast is greater than the ith probability thresholds	PODY_i	Categorical	Point-Stat Grid-Stat	PRC
Probability of false detection	POFD	Categorical	MODE Grid-Stat	MODE NBRCTCS
Probability of false detection when forecast is greater than the ith probability thresholds	POFD_i	Categorical	Point-Stat Grid-Stat	PRC
Pearson correlation coefficient	PR_CORR	Continuous	Ensemble-Stat Point-Stat Grid-Stat	SSVAR CNT
Rank of the observation	RANK	Ensemble	Ensemble-Stat	ORANK
Count of observations with the i-th rank	RANK_i	Ensemble	Ensemble-Stat	RHIST
Number of ranks used in computing Kendall's tau statistic	RANKS	Continuous	Point-Stat Grid-Stat	CNT
Refinement when forecast is between the ith and i+1th probability thresholds (repeated)	REFINEMENT_i	Probability	Point-Stat Grid-Stat	PJC
Reliability	RELIABILITY	Probability	Point-Stat Grid-Stat	PSTD
Number of times the i-th ensemble member's value was closest to the observation (repeated). When n members tie, 1/n is assigned to each member.	RELP_i	Ensemble	Ensemble-Stat	RELP
Resolution	RESOLUTION	Probability	Point-Stat Grid-Stat	PSTD

10.1.7 Statistics List S-T

Table 7: Statistics List S-T

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
S1 score	S1	Continuous	Grid-Stat	GRAD
S1 score with respect to observed gradient	S1_OG	Continuous	Grid-Stat	GRAD
Symmetric Extremal Dependency Index	SEDI	Categorical	Point-Stat Grid-Stat	CTS NBRCTS
Symmetric Extreme Dependency Score	SEDS	Categorical	Point-Stat Grid-Stat	CTS NBRCTS
Scatter Index	SI	Continuous	Point-Stat Grid-Stat	CNT
Spearman's rank correlation coefficient	SP_CORR	Continuous	Point-Stat Grid-Stat	CNT
Absolute value of SPEED_ERR	SPEED_ABSERR	Continuous	Point-Stat Grid-Stat	VCNT
Difference between the length of the average forecast wind vector and the average observed wind vector (in the sense F - O)	SPEED_ERR	Continuous	Point-Stat Grid-Stat	VCNT
Standard deviation of the mean of the UNPERTURBED ensemble	SPREAD	Ensemble	Ensemble-Stat	ECNT ORANK
Standard deviation of the mean of the PERTURBED ensemble	SPREAD_OERR	Ensemble	Ensemble-Stat	ECNT ORANK
Standard Deviation of unperturbed ensemble variance and the observation error variance	SPREAD_PLUS_OERR	Ensemble	Ensemble-Stat	ECNT ORANK
Track error of adeck relative to bdeck (nm)	TK_ERR	Continuous	TC-Pairs	PROBRIRW
Track error of adeck relative to bdeck (nm)	TK_ERR	Continuous	TC-Pairs	TCMPR

10.1.8 Statistics List U-Z

Table 8: Statistics List U-Z

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Mean U-component Forecast Anomaly	UFABAR	Continuous	Point-Stat Grid-Stat	VAL1L2
Mean U-component	UFBAR	Continuous	Point-Stat Grid-Stat	VL1L2
Uniform Fractions Skill Score	UFSS	Neighborhood	Grid-Stat	NBRCNT
Variability of Observations	UNCERTAINTY	Probability	Point-Stat Grid-Stat	PSTD
Mean U-component Observation Anomaly	UOABAR	Continuous	Point-Stat Grid-Stat	VAL1L2
Mean U-component Observation	UOBAR	Continuous	Point-Stat Grid-Stat	VL1L2
Mean U-component Squared Forecast Anomaly plus Squared Observation Anomaly	UVFFABAR	Continuous	Point-Stat Grid-Stat	VAL1L2
Mean U-component Squared Forecast plus Squared Observation	UVFFBAR	Continuous	Point-Stat Grid-Stat	VL1L2
Mean((uf-uc)*(uo-uc)+ (vf-vc)*(vo-vc))	UVFOABAR	Continuous	Point-Stat Grid-Stat	VAL1L2
Mean(uf*uo+vf*vo)	UVFOBAR	Continuous	Point-Stat Grid-Stat	VL1L2
Mean((uo-uc)^2+(vo-vc)^2)	UVOOABAR	Continuous	Point-Stat Grid-Stat	VAL1L2
Mean(uo^2+vo^2)	UVOOBAR	Continuous	Point-Stat Grid-Stat	VL1L2
Economic value of the base rate	VALUE_BASE	Probability	Point-Stat Grid-Stat	ECLV
Relative value for the ith Cost/Loss ratio	VALUE_i	Probability	Point-Stat Grid-Stat	ECLV
Maximum variance	VAR_MAX	Ensemble	Ensemble-Stat	SSVAR
Average variance	VAR_MEAN	Ensemble	Ensemble-Stat	SSVAR
Minimum variance	VAR_MIN	Ensemble	Ensemble-Stat	SSVAR
Direction of the vector difference between the average forecast and average wind vectors	VDIFF_DIR	Continuous	Point-Stat Grid-Stat	VCNT
Length (speed) of the vector difference between the average forecast and average observed wind vectors	VDIFF_SPEED	Continuous	Point-Stat Grid-Stat	VCNT
Mean(vf-vc)	VFABAR	Continuous	Point-Stat Grid-Stat	VAL1L2
Mean(vf)	VFBAR	Continuous	Point-Stat Grid-Stat	VL1L2
Mean(vo-vc)	VOABAR	Continuous	Point-Stat Grid-Stat	VAL1L2
Mean(vo)	VOBAR	Continuous	Point-Stat Grid-Stat	VL1L2

10.2 Diagnostics Database

10.2.1 Diagnostics List A-B

Table 9: Diagnostics List A-B

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Difference between the axis angles of two objects (in degrees)	ANGLE_DIFF	Diagnostic	MODE	MODE
Object area (in grid squares)	AREA	Diagnostic	MODE MTD	MODE obj
Forecast object area divided by the observation object area (unitless)	AREA_RATIO	Diagnostic	MODE	MODE obj
Area of the object that meet the object definition threshold criteria (in grid squares)	AREA_THRESH	Diagnostic	MODE	MODE obj
Absolute value of the difference between the aspect ratios of two objects (unitless)	ASPECT_DIFF	Diagnostic	MODE	MODE obj
Object axis angle (in degrees)	AXIS_ANG	Diagnostic	MODE MTD	MTD obj
Difference in spatial axis plane angles	AXIS_DIFF	Diagnostic	MTD	MTD obj
Blocking Index	Blocking Index	Diagnostic	METplus Use Case	n/a
Minimum distance between the boundaries of two objects	BOUNDARY _DIST	Diagnostic	MODE	MODE obj

10.2.2 Diagnostics List C-E

Table 10: Diagnostics List C-E

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Total great circle distance travelled by the 2D spatial centroid over the lifetime of the 3D object	CDIST_TRAVELLED	Diagnostic	MTD	MTD 3D obj
Distance between two objects centroids (in grid units)	CENTROID_DIST	Diagnostic	MODE	MODE obj
Latitude of centroid	CENTROID_LAT	Diagnostic	MTD MODE	MTD 2D & 3D obj MODE obj
Longitude of centroid	CENTROID_LON	Diagnostic	MTD MODE	MTD 2D & 3D obj MODE obj
Time coordinate of centroid	CENTROID_T	Diagnostic	MTD	MTD 3D obj
X coordinate of centroid	CENTROID_X	Diagnostic	MTD MODE	MTD 2D & 3D obj MODE obj
Y coordinate of centroid	CENTROID_Y	Diagnostic	MTD MODE	MTD 2D & 3D obj MODE obj
Space-Time Coherence Diagram	Coherence Diagram	Diagnostic	METplus Use Case	n/a
Ratio of the difference between the area of an object and the area of its convex hull divided by the area of the complex hull (unitless)	COMPLEXITY	Diagnostic	MODE	MODE obj
Ratio of complexities of two objects defined as the lesser of the forecast complexity divided by the observation complexity or its reciprocal (unitless)	COMPLEXITY_RATIO	Diagnostic	MODE	MODE obj
Minimum distance between the convex hulls of two objects (in grid units)	CONVEX_HULL_DIST	Diagnostic	MODE	MODE obj
Radius of curvature	CURVATURE	Diagnostic	MODE	MODE obj
Ratio of the curvature	CURVATURE_RATIO	Diagnostic	MODE	MODE obj
Center of curvature (in grid coordinates)	CURVATURE_X	Diagnostic	MODE	MODE obj
Center of curvature (in grid coordinates)	CURVATURE_Y	Diagnostic	MODE	MODE obj

10.2.3 Diagnostics List F

Table 11: Diagnostics List F

Statistics Long Name	METplus Name	S
Number of forecast clusters	FCST_CLUS	D
Number of points used to define the hull of all of the cluster forecast objects	FCST_CLUS _HULL	D
Forecast Cluster Convex Hull Point Latitude	FCST_CLUS _HULL_LAT	D
Forecast Cluster Convex Hull Point Longitude	FCST_CLUS _HULL _LON	D
Number of Forecast Cluster Convex Hull Points	FCST_CLUS _HULL_NPTS	D
Forecast Cluster Convex Hull Starting Index	FCST_CLUS _HULL_START	D
Forecast Cluster Convex Hull Point X-Coordinate	FCST_CLUS _HULL_X	D
Forecast Cluster Convex Hull Point Y-Coordinate	FCST_CLUS _HULL_Y	D
Forecast Object Raw Values	FCST_OBJ _RAW	D
Number of simple forecast objects	FCST_SIMP	D
Number of points used to define the boundaries of all of the simple forecast objects	FCST_SIMP _BDY	D
Forecast Simple Boundary Latitude	FCST_SIMP _BDY_LAT	D
Forecast Simple Boundary Longitude	FCST_SIMP _BDY_LON	D
Number of Forecast Simple Boundary Points	FCST_SIMP _BDY_NPTS	D
Forecast Simple Boundary Starting Index	FCST_SIMP _BDY_START	D
Forecast Simple Boundary X-Coordinate	FCST_SIMP _BDY_X	D
Forecast Simple Boundary Y-Coordinate	FCST_SIMP _BDY_Y	D
Number of points used to define the hull of all of the simple forecast objects	FCST_SIMP _HULL	D
Forecast Simple Convex Hull Point Latitude	FCST_SIMP _HULL_LAT	D
Forecast Simple Convex Hull Point Longitude	FCST_SIMP _HULL_LON	D
Number of Forecast Simple Convex Hull Points	FCST_SIMP _HULL_NPTS	D
Forecast Simple Convex Hull Starting Index	FCST_SIMP _HULL_START	D
Forecast Simple Convex Hull Point X-Coordinate	FCST_SIMP _HULL_X	D
Forecast Simple Convex Hull Point Y-Coordinate	FCST_SIMP _HULL_Y	D
Number of thresholds applied to the forecast	FCST _THRESH _LENGTH	D
Number of thresholds applied to the forecast	FCST_THRESH _LENGTH	D
Forecast energy squared for this scale	FENERGY	
Mean of absolute value of forecast gradients	FGBAR	
Ratio of forecast and observed gradients	FGOG_RATIO	
Pratt's Figure of Merit from observation to forecast	FOM_FO	D
Maximum of FOM_FO and FOM_OF	FOM_MAX	D
Mean of FOM_FO and FOM_OF	FOM_MEAN	D
Minimum of FOM_FO and FOM_OF	FOM_MIN	D
Pratt's Figure of Merit from forecast to observation	FOM_OF	D

10.2.4 Diagnostics List G-L

Table 12: Diagnostics List G-L

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Distance between the forecast and Best track genesis events (km)	GEN_DIST	Diagnostic	TC-Gen	GENMPR
Forecast minus Best track genesis time in HHMMSS format	GEN_TDIFF	Diagnostic	TC-Gen	GENMPR
Hausdorff Distance	HAUSDORFF	Diagnostic	GridStat	DMAP
Hovmoeller Diagram	Hovmoeller	Diagnostic	METplus Use Case	n/a
Best track genesis minus forecast initialization time in HHMMSS format	INIT_TDIFF	Diagnostic	TC-Gen	GENMPR
10th, 25th, 50th, 75th, 90th, and user-specified percentiles of intensity of the raw field within the object or time slice	INTENSITY _10, _25, _50, _75, _90, _NN	Diagnostic	MODE	MODE obj
Sum of the intensities of the raw field within the object (variable units)	INTENSITY _SUM	Diagnostic	MODE	MODE obj
Total interest for this object pair	INTEREST	Diagnostic	MTD MODE	MTD 3D obj MODE obj
Intersection area of two objects (in grid squares)	INTERSECTION_AREA	Diagnostic	MODE	MODE obj
Ratio of intersection area to the lesser of the forecast and observation object areas (unitless)	INTERSECTION_OVER_AREA	Diagnostic	MODE	MODE obj
"Volume" of object intersection	INTERSECTION_VOLUME	Diagnostic	MTD	MTD 3D obj
The intensity scale skill score	ISC		WaveletStat	ISC
The scale at which all information following applies	ISCALE		WaveletStat	ISC
Joint Probability Distribution between variable	Joint PDF to Diagnose Relationship	Diagnostic	GridDiag	n/a
Dimension of the latitude	LAT	Diagnostic	MODE	MODE obj
Length of the enclosing rectangle	LENGTH	Diagnostic	MODE	MODE obj
Dimension of the longitude	LON	Diagnostic	MODE	MODE obj

10.2.5 Diagnostics List M-O

Table 13: Diagnostics List M-O

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Meridional Means	Meridional Means	Diagnostic	METplus Use Case	n/a
Mean of maximum of absolute values of forecast and observed gradients	MGBAR		Grid-Stat	GRAD
Number of cluster objects	N_CLUS	Diagnostic	MODE	MODE obj
Number of simple forecast objects	N_FCST_SIMP	Diagnostic	MODE	MODE obj
Number of simple observation objects	N_OBS_SIMP	Diagnostic	MODE	MODE obj
Number of observed clusters	OBS_CLUS	Diagnostic	MODE	MODE obj
Number of points used to define the hull of all of the cluster observation objects	OBS_CLUS _HULL	Diagnostic	MODE	MODE obj
Observation Cluster Convex Hull Point Latitude	OBS_CLUS _HULL_LAT	Diagnostic	MODE	MODE obj
Observation Cluster Convex Hull Point Longitude	OBS_CLUS _HULL_LON	Diagnostic	MODE	MODE obj
Number of Observation Cluster Convex Hull Points	OBS_CLUS _HULL_NPTS	Diagnostic	MODE	MODE obj
Observation Cluster Convex Hull Starting Index	OBS_CLUS _HULL_START	Diagnostic	MODE	MODE obj
Observation Cluster Convex Hull Point X-Coordinate	OBS_CLUS _HULL_X	Diagnostic	MODE	MODE obj
Observation Cluster Convex Hull Point Y-Coordinate	OBS_CLUS _HULL_Y	Diagnostic	MODE	MODE obj
Number of simple observation objects	OBS_SIMP	Diagnostic	MODE	MODE obj
Number of points used to define the boundaries of the simple observation objects	OBS_SIMP _BDY	Diagnostic	MODE	MODE obj
Observation Simple Boundary Point Latitude	OBS_SIMP _BDY_LAT	Diagnostic	MODE	MODE obj
Observation Simple Boundary Point Longitude	OBS_SIMP _BDY_LON	Diagnostic	MODE	MODE obj
Number of Observation Simple Boundary Points	OBS_SIMP _BDY_NPTS	Diagnostic	MODE	MODE obj
Number of points used to define the hull of the simple observation objects	OBS_SIMP _HULL	Diagnostic	MODE	MODE obj
Number of Observation Simple Convex Hull Points	OBS_SIMP _HULL_NPTS	Diagnostic	MODE	MODE obj
Observed energy squared for this scale	OENERGY		Wavelet-Stat	ISC
Mean of absolute value of observed gradients	OGBAR		Grid-Stat	GRAD
OLR-based MJO Index	OMI	Diagnostic	METplus Use Case	n/a

10.2.6 Diagnostics List P-Z

Table 14: Diagnostics List P-Z

Statistics Long Name	METplus Name	Statistic Type	Tools	METplus Line Type
Ratio of the nth percentile (INTENSITY_NN column) of intensity of the two objects	PERCENTILE _INTENSITY _RATIO	Diagnostic	MODE	MODE obj
Phase Diagram for RMM and OMI	Phase Diagram	Diagnostic	METplus Use Case	n/a
Realtime Multivariate MJO Index	RMM	Diagnostic	METplus Use Case	n/a
Spatial distance between (x, y) coordinates of object spacetime centroid	SPACE _CENTROID _DIST	Diagnostic	MTD	MTD 3D obs
Difference in object speeds	SPEED_DELTA	Diagnostic	MTD	MTD 3D obs
Difference in object starting time steps	START_TIME _DELTA	Diagnostic	MTD	MTD 3D obj
Symmetric difference of two objects (in grid squares)	SYMMETRIC _DIFF	Diagnostic	MODE	MODE obj
Difference in t index of object spacetime centroid	TIME _CENTROID _DELTA	Diagnostic	MTD	MTD 3D obj
Union area of two objects (in grid squares)	UNION_AREA	Diagnostic	MODE	MODE obj
Integer count of the number of 3D "cells" in an object	VOLUME	Diagnostic	MTD	MTD 3D obj
Forecast object volume divided by observation object volume	VOLUME _RATIO	Diagnostic	MTD	MTD 3D obj
Weather Regime Index	Weather Regime Index	Diagnostic	METplus Use Case	n/a
Width of the enclosing rectangle (in grid units)	WIDTH	Diagnostic	MODE	MODE obj
X component of object velocity	X_DOT	Diagnostic	MTD	MTD 3D obj
X component position error (nm)	X_ERR	Diagnostic	TC-Pairs	PROBRIRW
X component position error (nm)	X_ERR	Diagnostic	TC-Pairs	TCMPR
y component of object velocity	Y_DOT	Diagnostic	MTD	MTD 3D obj
Y component position error (nm)	Y_ERR	Diagnostic	TC-Pairs	PROBRIRW TCMPR
Zonal Means	Zonal Means	Diagnostic	METplus Use Case	n/a
Zhu's Measure from observation to forecast	ZHU_FO	Diagnostic	Grid-Stat	DMAP
Maximum of ZHU_FO and ZHU_OF	ZHU_MAX	Diagnostic	Grid-Stat	DMAP
Mean of ZHU_FO and ZHU_OF	ZHU_MEAN	Diagnostic	Grid-Stat	DMAP
Minimum of ZHU_FO and ZHU_OF	ZHU_MIN	Diagnostic	Grid-Stat	DMAP

Bibliography

[Albertson1998] Alberson, S.D., 1998: Five-day Tropical cyclone track forecasts in the North Atlantic Basin. *Weather & Forecasting*, 13, 1005-1015.

[Bradley2008] Bradley, A.A., S.S. Schwartz, and T. Hashino, 2008: Sampling Uncertainty and Confidence Intervals for the Brier Score and Brier Skill Score. *Weather and Forecasting*, 23, 992-1006.

[Brill2009] Brill, K.F., and F. Mesinger, 2009: Applying a general analytic method for assessing bias sensitivity to bias-adjusted threat and equitable threat scores. *Weather and Forecasting*, 24, 17481754.

[Brown2007] Brown, B.G., R. Bullock, J. Halley Gotway, D. Ahijevych, C. Davis, E. Gilleland, and L. Holland, 2007: Application of the MODE object-based verification tool for the evaluation of model precipitation fields. *AMS 22nd Conference on Weather Analysis and Forecasting and 18th Conference on Numerical Weather Prediction*, 25-29 June, Park City, Utah, American Meteorological Society (Boston), Available at http://ams.confex.com/ams/pdfpapers/124856.pdf.

[Bullock2016] Bullock, R., T. Fowler, and B. Brown, 2016: Method for Object-Based Diagnostic Evaluation. NCAR Tech. Note NCAR/TN-532+STR, 66 pp.

[Candille2008] Candille, G., and O. Talagrand, 2008: Impact of observational error on the validation of ensemble prediction systems. *Q. J. R. Meteorol. Soc*. 134: 959971.

[Casati2004] Casati, B., G. Ross, and D. Stephenson, 2004: A new intensity-scale approach for the verification of spatial precipitation forecasts. *Meteorol. Appl*. 11, 141-154.

[Davis2006a] Davis, C.A., B.G. Brown, and R.G. Bullock, 2006a: Object-based verification of precipitation forecasts, Part I: Methodology and application to mesoscale rain areas. *Monthly Weather Review*, 134, 1772-1784.

[Davis2006b] Davis, C.A., B.G. Brown, and R.G. Bullock, 2006b: Object-based verification of precipitation forecasts, Part II: Application to convective rain systems. *Monthly Weather Review*, 134, 1785-1795.

[Dawid1984] Dawid, A.P., 1984: Statistical theory: The prequential approach. *J. Roy. Stat. Soc*, A147, 278-292.

[Ebert2008] Ebert, E.E., 2008: Fuzzy verification of high-resolution gridded forecasts: a review and proposed framework. *Meteorological Applications*, 15, 51-64.

[Eckel2012] Eckel, F.A., M.S. Allen, M.C. Sittel, 2012: Estimation of Ambiguity in Ensemble Forecasts. *Wea. Forecasting*, 27, 50-69. doi: http://dx.doi.org/10.1175/WAF-D-11-00015.1

[Efron2007] Efron, B. 2007: Correlation and large-scale significance testing. *Journal of the American Statistical Association*, 102(477), 93-103.

[Gilleland2010] Gilleland, E., 2010: Confidence intervals for forecast verification. *NCAR Technical Note* NCAR/TN-479+STR, 71pp.

[Gneiting2004] Gneiting, T., A. Westveld, A. Raferty, and T. Goldman, 2004: *Calibrated Probabilistic Forecasting Using Ensemble Model Output Statistics and Minimum CRPS Estimation*. Technical Report no. 449, Department of Statistics, University of Washington. [Available online at http://www.stat.washington.edu/www/research/reports/]

[Hamill2001] Hamill, T.M., 2001: Interpretation of rank histograms for verifying ensemble forecasts. *Mon. Wea. Rev*., 129, 550-560.

[Hogan2009] Hogan, R., E. O?Connor, and A. Illingworth, 2009: Verification of cloud-fraction forecasts. *Quart. Jour. Roy. Meteorol. Soc.*, 135, 1494-1511.

[Jolliffe2012] Jolliffe, I.T., and D.B. Stephenson, 2012: *Forecast verification. A practitioner?s guide in atmospheric science*. Wiley and Sons Ltd, 240 pp.

[Knaff2003] Knaff, J.A., M. DeMaria, C.R. Sampson, and J.M. Gross, 2003: Statistical, Five-Day Tropical Cyclone Intensity Forecasts Derived from Climatology and Persistence.?*Weather & Forecasting,*? Vol. 18 Issue 2, p. 80-92.

[Mason2004] Mason, S.J., 2004: On Using ?Climatology? as a Reference Strategy in the Brier and Ranked Probability Skill Scores. *Mon. Wea. Rev*., 132, 18911895.

[Mittermaier2013] Mittermaier, M., 2013: A strategy for verifying near-convection-resolving model forecasts at observing sites. Wea. Forecasting, 29, 185-204.

[Mood1974] Mood, A.M., F.A. Graybill and D.C. Boes, 1974: *Introduction to the Theory of Statistics*, McGraw-Hill, 299-338.

[Murphy1987] Murphy, A.H., and R.L. Winkler, 1987: A general framework for forecast verification. *Monthly Weather Review*, 115, 1330-1338.

[Roberts2008] Roberts, N.M., and H.W. Lean, 2008: Scale-selective verification of rainfall accumulations from high-resolution forecasts of convective events. *Monthly Weather Review*, 136, 78-97.

[Saetra2004] Saetra O., H. Hersbach, J-R Bidlot, D. Richardson, 2004: Effects of observation errors on the statistics for ensemble spread and reliability. *Mon. Weather Rev*. 132: 14871501.

[Santos2012] Santos C. and A. Ghelli, 2012: Observational probability method to assess ensemble precipitation forecasts. *Q. J. R. Meteorol. Soc.* 138: 209221.

[Stephenson2000] Stephenson, D.B., 2000: Use of the ?Odds Ratio? for diagnosing forecast skill. *Weather and Forecasting*, 15, 221-232.

[Stephenson2008] Stephenson, D.B., B. Casati, C.A.T. Ferro, and C.A. Wilson, 2008: The extreme dependency score: A non-vanishing measure for forecasts of rare events. *Meteor. Appl.* 15, 41-50.

[Weniger2016] Weniger, M., F. Kapp, and P. Friederichs, 2016: Spatial Verification Using Wavelet Transforms: A Review. *Quarterly Journal of the Royal Meteorological Society*, 143, 120-136.

[Wilks2010] Wilks, D.S. 2010: Sampling distributions of the Brier score and Brier skill score under serial dependence. Q.J.R. Meteorol. Soc., 136, 21092118. doi:10.1002/qj.709

[Wilks2011] Wilks, D., 2011: *Statistical methods in the atmospheric sciences*. Elsevier, San Diego.