METplus User's Guide

version 5.0.0-beta2-dev

Jul 11, 2022

Contents

1 Overview **3**

1.1 Purpose and organization of the User's Guide . . . 3
1.2 The Developmental Testbed Center (DTC) . . . 3
1.3 METplus Wrappers goals and design philosophy . . . 4
1.4 METplus Wrappers Components . . . 4
1.5 METplus Components Python Requirements . . . 4
1.6 METplus Release Notes . . . 5
1.6.1 METplus Components Release Note Links . . . 6
1.6.1.1 Release Notes - Latest Official Release . . . 6
1.6.1.2 Release Notes - Development Release . . . 6
1.6.2 METplus Wrappers Release Notes . . . 6
1.6.2.1 METplus Version 5.0.0 Beta 1 Release Notes (2022-06-22) . . . 6
1.7 Future development plans . . . 7
1.8 Code support . . . 7

2 Software Installation **9**

2.1 Introduction . . . 9
2.2 Supported architectures . . . 9
2.3 Programming/scripting languages . . . 9
2.4 Requirements . . . 9
2.4.1 Software Requirements . . . 9
2.4.2 Python Package Requirements . . . 10
2.5 Getting the METplus Wrappers source code . . . 11
2.5.1 Get the source code via Web Browser . . . 11
2.5.2 Get the source code via Command Line . . . 12
2.6 Obtain sample input data . . . 12
2.7 METplus Wrappers directory structure . . . 13
2.8 External Components . . . 14
2.8.1 GFDL Tracker (optional) . . . 14
2.9 Disable UserScript wrapper (optional) . . . 14
2.10 Add ush directory to shell path (optional) . . . 14
2.11 Set Default Configuration File for Shared Install . . . 15

3 System Configuration **17**

3.1 Config Best Practices / Recommendations . . . 17
3.2 Default Configuration File . . . 18
3.2.1 Required (/path/to) . . . 18
3.2.1.1 MET_INSTALL_DIR . . . 18
3.2.1.2 INPUT_BASE . . . 19
3.2.1.3 OUTPUT_BASE . . . 19
3.2.2 Optional . . . 19
3.2.2.1 MET_BIN_DIR . . . 19
3.2.2.2 METPLUS_CONF . . . 20
3.2.2.3 TMP_DIR . . . 20
3.2.2.4 STAGING_DIR . . . 20
3.2.2.5 OMP_NUM_THREADS . . . 20
3.2.2.6 CONVERT . . . 21
3.2.2.7 GEMPAKTOCF_JAR . . . 21
3.2.3 Logging . . . 21
3.2.3.1 Log File Information . . . 21
3.2.3.1.1 LOG_METPLUS . . . 21
3.2.3.1.2 LOG_DIR . . . 21
3.2.3.1.3 LOG_TIMESTAMP_TEMPLATE . . . 22
3.2.3.1.4 LOG_TIMESTAMP_USE_DATATIME . . . 22
3.2.3.1.5 LOG_MET_OUTPUT_TO_METPLUS . . . 22
3.2.3.2 Log Level Information . . . 22
3.2.3.2.1 LOG_LEVEL . . . 23
3.2.3.2.2 LOG_MET_VERBOSITY . . . 23
3.2.3.3 Log Formatting Information . . . 23
3.2.3.3.1 LOG_INFO_LINE_FORMAT . . . 24
3.2.3.3.2 LOG_ERR_LINE_FORMAT . . . 24
3.2.3.3.3 LOG_DEBUG_LINE_FORMAT . . . 24
3.2.3.3.4 LOG_LINE_DATE_FORMAT . . . 25
3.2.3.3.5 LOG_LINE_FORMAT . . . 25
3.3 User Configuration File . . . 25
3.4 Use Case Configuration Files . . . 26
3.5 Running METplus . . . 26
3.5.1 Example Wrapper Use Case . . . 26
3.5.2 GridStat Wrapper Basic Use Case . . . 27
3.6 Common Config Variables . . . 28
3.6.1 Timing Control . . . 28
3.6.1.1 LOOP_BY . . . 28
3.6.1.2 Looping by Valid Time . . . 28
3.6.1.2.1 VALID_TIME_FMT . . . 28
3.6.1.2.2 VALID_BEG . . . 28
3.6.1.2.3 VALID_END . . . 29
3.6.1.2.4 VALID_INCREMENT . . . 29
3.6.1.2.5 VALID_LIST . . . 30
3.6.1.3 Looping by Initialization Time . . . 30
3.6.1.3.1 INIT_TIME_FMT . . . 30
3.6.1.3.2 INIT_BEG . . . 30
3.6.1.3.3 INIT_END . . . 30

3.6.1.3.4 INIT_INCREMENT . . . 31
3.6.1.3.5 INIT_LIST . . . 31
3.6.1.4 Looping over Forecast Leads . . . 31
3.6.1.4.1 LEAD_SEQ . . . 32
3.6.1.4.2 INIT_SEQ . . . 33
3.6.1.5 Time Interval Units . . . 33
3.6.1.6 Skipping Times . . . 34
3.6.1.7 Realtime Looping . . . 35
3.6.1.7.1 Now and Today . . . 35
3.6.1.7.2 Shift Keyword . . . 35
3.6.1.7.3 Truncate Keyword . . . 36
3.6.2 Process List . . . 37
3.6.3 Loop Order . . . 37
3.6.4 Custom Looping . . . 39
3.6.5 Field Info . . . 40
3.6.5.1 FCST_VAR<n>_NAME . . . 40
3.6.5.2 FCST_VAR<n>_LEVELS . . . 40
3.6.5.3 OBS_VAR<n>_NAME . . . 41
3.6.5.4 OBS_VAR<n>_LEVELS . . . 41
3.6.5.5 Read explicit time dimension from a NetCDF level . . . 43
3.6.5.6 Substituting Current Level . . . 43
3.6.5.7 FCST_VAR<n>_THRESH / OBS_VAR<n>_THRESH . . . 44
3.6.5.8 FCST_VAR<n>_OPTIONS / OBS_VAR<n>_OPTIONS . . . 44
3.6.5.9 Probabilistic Forecast Fields . . . 45
3.6.5.10 Wrapper Specific Field Info . . . 46
3.6.6 Directory and Filename Template Info . . . 47
3.6.6.1 Using Templates to find Observation Data . . . 47
3.6.6.2 Using Templates to find Forecast Data . . . 48
3.6.6.3 Using Templates to find Data Assimilation Data . . . 48
3.6.6.4 Shifting Times in Filename Templates . . . 49
3.6.6.5 Using Windows to find Valid Files . . . 49
3.6.6.6 Wrapper Specific Windows . . . 50
3.6.7 Runtime Frequency . . . 51
3.7 How METplus controls MET configuration variables . . . 54
3.7.1 GridStat Simple Example . . . 54
3.7.2 GridStat Dictionary example . . . 55
3.7.3 GridStat Fields . . . 57
3.8 Reconcile Default Values . . . 57
3.8.1 EnsembleStatConfig . . . 57
3.8.1.1 message_type . . . 57
3.8.1.2 climo_cdf.cdf_bins . . . 58
3.8.1.3 mask.poly . . . 59
3.8.1.4 output_flag (multiple items) . . . 60
3.8.2 GridStatConfig . . . 60
3.8.2.1 cat_thresh . . . 60
3.8.2.2 output_flag (multiple items) . . . 61
3.8.2.3 nc_pairs_flag (multiple items) . . . 62
3.8.3 MODEConfig . . . 62

3.8.3.1 grid_res . . . 62
3.8.3.2 fcst.merge_thresh and fcst.merge_flag . . . 63
3.8.3.3 fcst_raw_plot.color_table . . . 63
3.8.3.4 obs_raw_plot.color_table . . . 64
3.8.3.5 mask_missing_flag . . . 64
3.8.4 PB2NCConfig . . . 64
3.8.4.1 level_category . . . 64
3.8.4.2 quality_mark_thresh . . . 64
3.8.4.3 time_summary.step and time_summary.width . . . 65
3.8.4.4 pb_report_type . . . 65
3.8.5 PointStatConfig . . . 66
3.8.5.1 regrid.method and regrid_width . . . 66
3.8.5.2 obs_quality . . . 66
3.8.5.3 climo_mean.time_interp_method and climo_stdev.time_interp_method . . . 67
3.8.5.4 interp.type.method and interp.type.width . . . 68
3.9 Overriding Unsupported MET configuration variables . . . 68
3.9.1 MET Config Override GridStat Simple Example . . . 69
3.10 User Environment Variables . . . 70
3.11 Setting Config Variables with Environment Variables . . . 71
3.12 Updating Configuration Files - Handling Deprecated Configuration Variables . . . 71
3.12.1 Simple Rename . . . 72
3.12.2 FCST/OBS/BOTH Variables . . . 72
3.12.3 PCPCombine Input Levels . . . 73
3.12.4 MET Configuration Files . . . 73
3.12.5 SED Commands . . . 74
3.12.6 Validate Config Helper Script . . . 75

4 Python Wrappers **79**
4.1 ASCII2NC . . . 79
4.1.1 Description . . . 79
4.1.2 METplus Configuration . . . 79
4.1.3 MET Configuration . . . 80
4.2 CyclonePlotter . . . 82
4.2.1 Description . . . 82
4.2.2 METplus Configuration . . . 82
4.3 EnsembleStat . . . 83
4.3.1 Description . . . 83
4.3.2 METplus Configuration . . . 83
4.3.3 MET Configuration . . . 87
4.4 Example . . . 99
4.4.1 Description . . . 99
4.4.2 Configuration . . . 100
4.5 ExtractTiles . . . 100
4.5.1 Description . . . 100
4.5.2 METplus Configuration . . . 100
4.6 GempakToCF . . . 101
4.6.1 Description . . . 101
4.6.2 METplus Configuration . . . 101

4.7 GenEnsProd 102
4.7.1 Description 102
4.7.2 METplus Configuration 102
4.7.3 MET Configuration 104
4.8 GenVxMask 110
4.8.1 Description 110
4.8.2 Configuration 111
4.9 GFDLTracker 111
4.9.1 Description 111
4.9.2 METplus Configuration 111
4.9.3 NML Configuration 114
4.10 GridDiag 127
4.10.1 Description 127
4.10.2 METplus Configuration 128
4.10.3 MET Configuration 128
4.11 GridStat 131
4.11.1 Description 131
4.11.2 METplus Configuration 131
4.11.3 MET Configuration 135
4.12 IODA2NC 145
4.12.1 Description 145
4.12.2 METplus Configuration 145
4.12.3 MET Configuration 146
4.13 MakePlots 151
4.13.1 Description 151
4.13.2 METplus Configuration 151
4.14 METdbLoad 153
4.14.1 Description 153
4.14.2 METplus Configuration 153
4.14.3 XML Configuration 154
4.15 MODE 156
4.15.1 Description 156
4.15.2 METplus Configuration 156
4.15.3 MET Configuration 159
4.16 MTD 171
4.16.1 Description 171
4.16.2 METplus Configuration 171
4.16.3 MET Configuration 172
4.17 PB2NC 180
4.17.1 Description 180
4.17.2 METplus Configuration 180
4.17.3 MET Configuration 181
4.18 PCPCombine 186
4.18.1 Description 186
4.18.2 METplus Configuration 186
4.19 PlotDataPlane 189
4.19.1 Description 189
4.19.2 Configuration 189

4.20 Point2Grid . . . 189
4.20.1 Description . . . 189
4.20.2 METplus Configuration . . . 189
4.21 PointStat . . . 190
4.21.1 Description . . . 190
4.21.2 Configuration . . . 190
4.21.3 MET Configuration . . . 194
4.22 PyEmbedIngest . . . 202
4.22.1 Description . . . 202
4.22.2 METplus Configuration . . . 203
4.23 RegridDataPlane . . . 203
4.23.1 Description . . . 203
4.23.2 METplus Configuration . . . 203
4.24 SeriesAnalysis . . . 204
4.24.1 Description . . . 204
4.24.2 METplus Configuration . . . 205
4.24.3 MET Configuration . . . 207
4.25 SeriesByInit . . . 214
4.25.1 Description . . . 214
4.26 SeriesByLead . . . 214
4.26.1 Description . . . 214
4.27 StatAnalysis . . . 214
4.27.1 Description . . . 214
4.27.2 METplus Configuration . . . 215
4.27.3 MET Configuration . . . 218
4.28 TCGen . . . 224
4.28.1 Description . . . 224
4.28.2 METplus Configuration . . . 225
4.28.3 MET Configuration . . . 227
4.29 TCMPRPlotter . . . 239
4.29.1 Description . . . 239
4.29.2 METplus Configuration . . . 239
4.30 TCPairs . . . 241
4.30.1 Description . . . 241
4.30.2 METplus Configuration . . . 241
4.30.3 MET Configuration . . . 243
4.31 TCRMW . . . 249
4.31.1 Description . . . 249
4.31.2 METplus Configuration . . . 249
4.31.3 MET Configuration . . . 250
4.32 TCStat . . . 255
4.32.1 Description . . . 255
4.32.2 METplus Configuration . . . 255
4.32.3 MET Configuration . . . 256
4.33 UserScript . . . 265
4.33.1 Description . . . 265
4.33.2 METplus Configuration . . . 265

5 METplus Use Cases **267**
5.1 MET tools . . . 267
5.1.1 ASCII2NC . . . 267
5.1.1.1 ASCII2NC: Basic Use Case . . . 267
5.1.1.1.1 Scientific Objective . . . 267
5.1.1.1.2 Datasets . . . 267
5.1.1.1.3 METplus Components . . . 268
5.1.1.1.4 METplus Workflow . . . 268
5.1.1.1.5 METplus Configuration . . . 268
5.1.1.1.6 MET Configuration . . . 271
5.1.1.1.7 Running METplus . . . 272
5.1.1.1.8 Expected Output . . . 273
5.1.1.1.9 Keywords . . . 273
5.1.1.2 ASCII2NC: Using Python Embedding . . . 273
5.1.1.2.1 Scientific Objective . . . 273
5.1.1.2.2 Datasets . . . 273
5.1.1.2.3 METplus Components . . . 274
5.1.1.2.4 METplus Workflow . . . 274
5.1.1.2.5 METplus Configuration . . . 274
5.1.1.2.6 MET Configuration . . . 277
5.1.1.2.7 Python Embedding . . . 277
5.1.1.2.8 Running METplus . . . 277
5.1.1.2.9 Expected Output . . . 278
5.1.1.2.10 Keywords . . . 278
5.1.2 Cyclone Plotter . . . 278
5.1.2.1 CyclonePlotter: Basic Use Case . . . 278
5.1.2.1.1 Scientific Objective . . . 279
5.1.2.1.2 Datasets . . . 279
5.1.2.1.3 External Dependencies . . . 279
5.1.2.1.4 METplus Components . . . 279
5.1.2.1.5 METplus Workflow . . . 279
5.1.2.1.6 METplus Configuration . . . 279
5.1.2.1.7 MET Configuration . . . 281
5.1.2.1.8 Running METplus . . . 281
5.1.2.1.9 Expected Output . . . 282
5.1.2.1.10 Keywords . . . 282
5.1.3 EnsembleStat . . . 283
5.1.3.1 EnsembleStat: Using Python Embedding . . . 283
5.1.3.1.1 Scientific Objective . . . 283
5.1.3.1.2 Datasets . . . 283
5.1.3.1.3 METplus Components . . . 283
5.1.3.1.4 METplus Workflow . . . 283
5.1.3.1.5 METplus Configuration . . . 284
5.1.3.1.6 MET Configuration . . . 288
5.1.3.1.7 Python Embedding . . . 294
5.1.3.1.8 Running METplus . . . 294
5.1.3.1.9 Expected Output . . . 294
5.1.3.1.10 Keywords . . . 295

5.1.3.2 EnsembleStat: Basic Use Case . . . 295
5.1.3.2.1 Scientific Objective . . . 295
5.1.3.2.2 Datasets . . . 295
5.1.3.2.3 METplus Components . . . 296
5.1.3.2.4 METplus Workflow . . . 296
5.1.3.2.5 METplus Configuration . . . 296
5.1.3.2.6 MET Configuration . . . 302
5.1.3.2.7 Running METplus . . . 308
5.1.3.2.8 Expected Output . . . 308
5.1.3.2.9 Keywords . . . 309
5.1.4 Example . . . 309
5.1.4.1 Example: Introductory Use Case . . . 309
5.1.4.1.1 Scientific Objective . . . 309
5.1.4.1.2 Datasets . . . 309
5.1.4.1.3 METplus Components . . . 309
5.1.4.1.4 METplus Workflow . . . 310
5.1.4.1.5 METplus Configuration . . . 312
5.1.4.1.6 MET Configuration . . . 314
5.1.4.1.7 Running METplus . . . 314
5.1.4.1.8 Expected Output . . . 315
5.1.4.1.9 Keywords . . . 316
5.1.5 ExtractTiles . . . 316
5.1.5.1 ExtractTiles: Basic Use Case . . . 316
5.1.5.1.1 Scientific Objective . . . 316
5.1.5.1.2 Datasets . . . 316
5.1.5.1.3 METplus Components . . . 317
5.1.5.1.4 METplus Workflow . . . 317
5.1.5.1.5 METplus Configuration . . . 317
5.1.5.1.6 MET Configuration . . . 319
5.1.5.1.7 Running METplus . . . 319
5.1.5.1.8 Expected Output . . . 319
5.1.5.1.9 Keywords . . . 324
5.1.5.2 ExtractTiles: MTD Input . . . 324
5.1.5.2.1 Scientific Objective . . . 324
5.1.5.2.2 Datasets . . . 324
5.1.5.2.3 METplus Components . . . 325
5.1.5.2.4 METplus Workflow . . . 325
5.1.5.2.5 METplus Configuration . . . 325
5.1.5.2.6 MET Configuration . . . 326
5.1.5.2.7 Running METplus . . . 327
5.1.5.2.8 Expected Output . . . 327
5.1.5.2.9 Keywords . . . 328
5.1.6 GFDLTracker . . . 328
5.1.6.1 GFDLTracker: Extra Tropical Cyclone Use Case . . . 328
5.1.6.1.1 Scientific Objective . . . 328
5.1.6.1.2 Datasets . . . 328
5.1.6.1.3 METplus Components . . . 329
5.1.6.1.4 METplus Workflow . . . 329

5.1.6.1.5 METplus Configuration . . . 329
5.1.6.1.6 GFDL Tracker Configuration . . . 332
5.1.6.1.7 Running METplus . . . 335
5.1.6.1.8 Expected Output . . . 335
5.1.6.1.9 Keywords . . . 336
5.1.6.2 GFDLTracker: TC Genesis Use Case . . . 336
5.1.6.2.1 Scientific Objective . . . 336
5.1.6.2.2 Datasets . . . 336
5.1.6.2.3 METplus Components . . . 336
5.1.6.2.4 METplus Workflow . . . 337
5.1.6.2.5 METplus Configuration . . . 337
5.1.6.2.6 GFDL Tracker Configuration . . . 340
5.1.6.2.7 Running METplus . . . 343
5.1.6.2.8 Expected Output . . . 343
5.1.6.2.9 Keywords . . . 344
5.1.6.3 GFDLTracker: Tropical Cyclone Use Case . . . 344
5.1.6.3.1 Scientific Objective . . . 344
5.1.6.3.2 Datasets . . . 344
5.1.6.3.3 METplus Components . . . 344
5.1.6.3.4 METplus Workflow . . . 345
5.1.6.3.5 METplus Configuration . . . 345
5.1.6.3.6 GFDL Tracker Configuration . . . 348
5.1.6.3.7 Running METplus . . . 351
5.1.6.3.8 Expected Output . . . 351
5.1.6.3.9 Keywords . . . 351
5.1.7 GempakToCF . . . 352
5.1.7.1 GempakToCF: Basic Use Case . . . 352
5.1.7.1.1 Scientific Objective . . . 352
5.1.7.1.2 Datasets . . . 352
5.1.7.1.3 External Dependencies . . . 352
5.1.7.1.4 METplus Components . . . 353
5.1.7.1.5 METplus Workflow . . . 353
5.1.7.1.6 METplus Configuration . . . 353
5.1.7.1.7 Running METplus . . . 355
5.1.7.1.8 Expected Output . . . 355
5.1.7.1.9 Keywords . . . 356
5.1.8 GenEnsProd . . . 356
5.1.8.1 GenEnsProd: Basic Use Case . . . 356
5.1.8.1.1 Scientific Objective . . . 356
5.1.8.1.2 Datasets . . . 357
5.1.8.1.3 METplus Components . . . 357
5.1.8.1.4 METplus Workflow . . . 357
5.1.8.1.5 METplus Configuration . . . 357
5.1.8.1.6 MET Configuration . . . 361
5.1.8.1.7 Running METplus . . . 363
5.1.8.1.8 Expected Output . . . 364
5.1.8.1.9 Keywords . . . 364
5.1.9 GenVxMask . . . 365

5.1.9.1 GenVxMask: Multiple Masks . . . 365
5.1.9.1.1 Scientific Objective . . . 365
5.1.9.1.2 Datasets . . . 365
5.1.9.1.3 METplus Components . . . 365
5.1.9.1.4 METplus Workflow . . . 365
5.1.9.1.5 METplus Configuration . . . 366
5.1.9.1.6 MET Configuration . . . 368
5.1.9.1.7 Running METplus . . . 368
5.1.9.1.8 Expected Output . . . 368
5.1.9.1.9 Keywords . . . 369
5.1.9.2 GenVxMask: Using Arguments . . . 369
5.1.9.2.1 Scientific Objective . . . 369
5.1.9.2.2 Datasets . . . 369
5.1.9.2.3 METplus Components . . . 370
5.1.9.2.4 METplus Workflow . . . 370
5.1.9.2.5 METplus Configuration . . . 370
5.1.9.2.6 MET Configuration . . . 372
5.1.9.2.7 Running METplus . . . 372
5.1.9.2.8 Expected Output . . . 373
5.1.9.2.9 Keywords . . . 373
5.1.9.3 GenVxMask: Basic Use Case . . . 373
5.1.9.3.1 Scientific Objective . . . 374
5.1.9.3.2 Datasets . . . 374
5.1.9.3.3 METplus Components . . . 374
5.1.9.3.4 METplus Workflow . . . 374
5.1.9.3.5 METplus Configuration . . . 375
5.1.9.3.6 MET Configuration . . . 377
5.1.9.3.7 Running METplus . . . 377
5.1.9.3.8 Expected Output . . . 377
5.1.9.3.9 Keywords . . . 378
5.1.10 GridDiag . . . 378
5.1.10.1 GridDiag: Basic Use Case . . . 378
5.1.10.1.1 Scientific Objective . . . 378
5.1.10.1.2 Datasets . . . 378
5.1.10.1.3 METplus Components . . . 379
5.1.10.1.4 METplus Workflow . . . 379
5.1.10.1.5 METplus Configuration . . . 379
5.1.10.1.6 MET Configuration . . . 381
5.1.10.1.7 Running METplus . . . 382
5.1.10.1.8 Expected Output . . . 383
5.1.10.1.9 Keywords . . . 383
5.1.11 GridStat . . . 383
5.1.11.1 GridStat: Basic Use Case . . . 383
5.1.11.1.1 Scientific Objective . . . 383
5.1.11.1.2 Datasets . . . 384
5.1.11.1.3 METplus Components . . . 384
5.1.11.1.4 METplus Workflow . . . 384
5.1.11.1.5 METplus Configuration . . . 384

5.1.11.1.6 MET Configuration . . . 389
5.1.11.1.7 Running METplus . . . 393
5.1.11.1.8 Expected Output . . . 394
5.1.11.1.9 Keywords . . . 394
5.1.11.2 GridStat: Using Python Embedding . . . 394
5.1.11.2.1 Scientific Objective . . . 395
5.1.11.2.2 Datasets . . . 395
5.1.11.2.3 METplus Components . . . 395
5.1.11.2.4 METplus Workflow . . . 395
5.1.11.2.5 METplus Configuration . . . 395
5.1.11.2.6 MET Configuration . . . 399
5.1.11.2.7 Python Embedding . . . 404
5.1.11.2.8 Running METplus . . . 404
5.1.11.2.9 Expected Output . . . 405
5.1.11.2.10Keywords . . . 405
5.1.11.3 GridStat: Multiple Config Files Use Case . . . 405
5.1.11.3.1 Scientific Objective . . . 406
5.1.11.3.2 Datasets . . . 406
5.1.11.3.3 METplus Components . . . 406
5.1.11.3.4 METplus Workflow . . . 406
5.1.11.3.5 METplus Configuration . . . 407
5.1.11.3.6 MET Configuration . . . 413
5.1.11.3.7 Running METplus . . . 418
5.1.11.3.8 Expected Output . . . 419
5.1.11.3.9 Keywords . . . 419
5.1.12 IODA2NC . . . 419
5.1.12.1 IODA2NC: Basic Use Case . . . 419
5.1.12.1.1 Scientific Objective . . . 419
5.1.12.1.2 Datasets . . . 420
5.1.12.1.3 METplus Components . . . 420
5.1.12.1.4 METplus Workflow . . . 420
5.1.12.1.5 METplus Configuration . . . 420
5.1.12.1.6 MET Configuration . . . 422
5.1.12.1.7 Running METplus . . . 425
5.1.12.1.8 Expected Output . . . 425
5.1.12.1.9 Keywords . . . 425
5.1.13 METdbLoad . . . 426
5.1.13.1 METdbLoad: Basic Use Case . . . 426
5.1.13.1.1 Scientific Objective . . . 426
5.1.13.1.2 Datasets . . . 426
5.1.13.1.3 METplus Components . . . 426
5.1.13.1.4 METplus Workflow . . . 426
5.1.13.1.5 METplus Configuration . . . 426
5.1.13.1.6 XML Configuration . . . 428
5.1.13.1.7 Running METplus . . . 428
5.1.13.1.8 Expected Output . . . 429
5.1.13.1.9 Keywords . . . 429
5.1.14 MODE . . . 430

5.1.14.1 MODE: Using Python Embedding . . . 430
5.1.14.1.1 Scientific Objective . . . 430
5.1.14.1.2 Datasets . . . 430
5.1.14.1.3 METplus Components . . . 430
5.1.14.1.4 METplus Workflow . . . 430
5.1.14.1.5 METplus Configuration . . . 431
5.1.14.1.6 MET Configuration . . . 434
5.1.14.1.7 Python Embedding . . . 440
5.1.14.1.8 Running METplus . . . 440
5.1.14.1.9 Expected Output . . . 441
5.1.14.1.10Keywords . . . 441
5.1.14.2 MODE: Basic Use Case . . . 442
5.1.14.2.1 Scientific Objective . . . 442
5.1.14.2.2 Datasets . . . 442
5.1.14.2.3 METplus Components . . . 442
5.1.14.2.4 METplus Workflow . . . 442
5.1.14.2.5 METplus Configuration . . . 443
5.1.14.2.6 MET Configuration . . . 446
5.1.14.2.7 Running METplus . . . 452
5.1.14.2.8 Expected Output . . . 452
5.1.14.2.9 Keywords . . . 453
5.1.15 MTD . . . 453
5.1.15.1 MTD using Python Embedding . . . 453
5.1.15.1.1 Scientific Objective . . . 453
5.1.15.1.2 Datasets . . . 453
5.1.15.1.3 METplus Components . . . 454
5.1.15.1.4 METplus Workflow . . . 454
5.1.15.1.5 METplus Configuration . . . 454
5.1.15.1.6 MET Configuration . . . 457
5.1.15.1.7 Python Embedding . . . 463
5.1.15.1.8 Running METplus . . . 463
5.1.15.1.9 Expected Output . . . 464
5.1.15.1.10Keywords . . . 464
5.1.15.2 Basic MTD Use Case . . . 464
5.1.15.2.1 Scientific Objective . . . 465
5.1.15.2.2 Datasets . . . 465
5.1.15.2.3 METplus Components . . . 465
5.1.15.2.4 METplus Workflow . . . 465
5.1.15.2.5 METplus Configuration . . . 465
5.1.15.2.6 MET Configuration . . . 468
5.1.15.2.7 Running METplus . . . 474
5.1.15.2.8 Expected Output . . . 475
5.1.15.2.9 Keywords . . . 475
5.1.16 PB2NC . . . 475
5.1.16.1 PB2NC: Basic Use Case . . . 475
5.1.16.1.1 Scientific Objective . . . 476
5.1.16.1.2 Datasets . . . 476
5.1.16.1.3 METplus Components . . . 476

5.1.16.1.4 METplus Workflow . . . 476
5.1.16.1.5 METplus Configuration . . . 476
5.1.16.1.6 MET Configuration . . . 478
5.1.16.1.7 Running METplus . . . 481
5.1.16.1.8 Expected Output . . . 482
5.1.16.1.9 Keywords . . . 482
5.1.17 PCPCombine . . . 482
5.1.17.1 PCPCombine: Custom String Looping Use Case . . . 482
5.1.17.1.1 Scientific Objective . . . 483
5.1.17.1.2 Datasets . . . 483
5.1.17.1.3 METplus Components . . . 483
5.1.17.1.4 METplus Workflow . . . 483
5.1.17.1.5 METplus Configuration . . . 484
5.1.17.1.6 MET Configuration . . . 485
5.1.17.1.7 Running METplus . . . 485
5.1.17.1.8 Expected Output . . . 486
5.1.17.1.9 Keywords . . . 486
5.1.17.2 PCPCombine: Bucket Interval Use Case . . . 486
5.1.17.2.1 Scientific Objective . . . 487
5.1.17.2.2 Datasets . . . 487
5.1.17.2.3 METplus Components . . . 487
5.1.17.2.4 METplus Workflow . . . 487
5.1.17.2.5 METplus Configuration . . . 488
5.1.17.2.6 MET Configuration . . . 489
5.1.17.2.7 Running METplus . . . 489
5.1.17.2.8 Expected Output . . . 489
5.1.17.2.9 Keywords . . . 490
5.1.17.3 PCPCombine: User-defined Command Use Case . . . 490
5.1.17.3.1 Scientific Objective . . . 490
5.1.17.3.2 Datasets . . . 490
5.1.17.3.3 METplus Components . . . 491
5.1.17.3.4 METplus Workflow . . . 491
5.1.17.3.5 METplus Configuration . . . 491
5.1.17.3.6 MET Configuration . . . 492
5.1.17.3.7 Running METplus . . . 492
5.1.17.3.8 Expected Output . . . 493
5.1.17.3.9 Keywords . . . 493
5.1.17.4 PCPCombine: ADD Use Case . . . 494
5.1.17.4.1 Scientific Objective . . . 494
5.1.17.4.2 Datasets . . . 494
5.1.17.4.3 METplus Components . . . 494
5.1.17.4.4 METplus Workflow . . . 494
5.1.17.4.5 METplus Configuration . . . 495
5.1.17.4.6 MET Configuration . . . 496
5.1.17.4.7 Running METplus . . . 496
5.1.17.4.8 Expected Output . . . 496
5.1.17.4.9 Keywords . . . 497
5.1.17.5 PCPCombine: Python Embedding Use Case . . . 497

5.1.17.5.1 Scientific Objective . . . 497
5.1.17.5.2 Datasets . . . 497
5.1.17.5.3 External Dependencies . . . 498
5.1.17.5.4 METplus Components . . . 498
5.1.17.5.5 METplus Workflow . . . 498
5.1.17.5.6 METplus Configuration . . . 498
5.1.17.5.7 MET Configuration . . . 499
5.1.17.5.8 Running METplus . . . 500
5.1.17.5.9 Expected Output . . . 500
5.1.17.5.10 Keywords . . . 501
5.1.17.6 PCPCombine: SUM Use Case . . . 501
5.1.17.6.1 Scientific Objective . . . 501
5.1.17.6.2 Datasets . . . 501
5.1.17.6.3 METplus Components . . . 502
5.1.17.6.4 METplus Workflow . . . 502
5.1.17.6.5 METplus Configuration . . . 502
5.1.17.6.6 MET Configuration . . . 503
5.1.17.6.7 Running METplus . . . 503
5.1.17.6.8 Expected Output . . . 504
5.1.17.6.9 Keywords . . . 504
5.1.17.7 PCPCombine: SUBTRACT Use Case . . . 504
5.1.17.7.1 Scientific Objective . . . 504
5.1.17.7.2 Datasets . . . 504
5.1.17.7.3 METplus Components . . . 505
5.1.17.7.4 METplus Workflow . . . 505
5.1.17.7.5 METplus Configuration . . . 505
5.1.17.7.6 MET Configuration . . . 506
5.1.17.7.7 Running METplus . . . 506
5.1.17.7.8 Expected Output . . . 507
5.1.17.7.9 Keywords . . . 507
5.1.17.8 PCPCombine: DERIVE Use Case . . . 508
5.1.17.8.1 Scientific Objective . . . 508
5.1.17.8.2 Datasets . . . 508
5.1.17.8.3 METplus Components . . . 508
5.1.17.8.4 METplus Workflow . . . 508
5.1.17.8.5 METplus Configuration . . . 509
5.1.17.8.6 MET Configuration . . . 510
5.1.17.8.7 Running METplus . . . 510
5.1.17.8.8 Expected Output . . . 511
5.1.17.8.9 Keywords . . . 511
5.1.18 PlotDataPlane . . . 511
5.1.18.1 PlotDataPlane: Python Embedding Input . . . 511
5.1.18.1.1 Scientific Objective . . . 511
5.1.18.1.2 Datasets . . . 511
5.1.18.1.3 METplus Components . . . 512
5.1.18.1.4 METplus Workflow . . . 512
5.1.18.1.5 METplus Configuration . . . 512
5.1.18.1.6 MET Configuration . . . 514

5.1.18.1.7 Python Embedding . . . 514
5.1.18.1.8 Running METplus . . . 514
5.1.18.1.9 Expected Output . . . 515
5.1.18.1.10Keywords . . . 515
5.1.18.2 PlotDataPlane: NetCDF Input . . . 515
5.1.18.2.1 Scientific Objective . . . 515
5.1.18.2.2 Datasets . . . 515
5.1.18.2.3 METplus Components . . . 516
5.1.18.2.4 METplus Workflow . . . 516
5.1.18.2.5 METplus Configuration . . . 516
5.1.18.2.6 MET Configuration . . . 518
5.1.18.2.7 Running METplus . . . 518
5.1.18.2.8 Expected Output . . . 518
5.1.18.2.9 Keywords . . . 519
5.1.18.3 PlotDataPlane: GRIB1 Input . . . 519
5.1.18.3.1 Scientific Objective . . . 519
5.1.18.3.2 Datasets . . . 519
5.1.18.3.3 METplus Components . . . 520
5.1.18.3.4 METplus Workflow . . . 520
5.1.18.3.5 METplus Configuration . . . 520
5.1.18.3.6 MET Configuration . . . 522
5.1.18.3.7 Running METplus . . . 522
5.1.18.3.8 Expected Output . . . 522
5.1.18.3.9 Keywords . . . 523
5.1.19 Point2Grid . . . 523
5.1.19.1 Point2Grid: Basic Use Case . . . 523
5.1.19.1.1 Scientific Objective . . . 523
5.1.19.1.2 Datasets . . . 523
5.1.19.1.3 METplus Components . . . 524
5.1.19.1.4 METplus Workflow . . . 524
5.1.19.1.5 METplus Configuration . . . 524
5.1.19.1.6 MET Configuration . . . 527
5.1.19.1.7 Running METplus . . . 527
5.1.19.1.8 Expected Output . . . 527
5.1.19.1.9 Keywords . . . 528
5.1.20 PointStat . . . 528
5.1.20.1 PointStat: Basic Use Case . . . 528
5.1.20.1.1 Scientific Objective . . . 528
5.1.20.1.2 Datasets . . . 528
5.1.20.1.3 METplus Components . . . 529
5.1.20.1.4 METplus Workflow . . . 529
5.1.20.1.5 METplus Configuration . . . 529
5.1.20.1.6 MET Configuration . . . 532
5.1.20.1.7 Running METplus . . . 537
5.1.20.1.8 Expected Output . . . 537
5.1.20.1.9 Keywords . . . 538
5.1.20.2 PointStat: Using Python Embedding . . . 538
5.1.20.2.1 Scientific Objective . . . 538

5.1.20.2.2 Datasets . . . 538
5.1.20.2.3 METplus Components . . . 539
5.1.20.2.4 METplus Workflow . . . 539
5.1.20.2.5 METplus Configuration . . . 539
5.1.20.2.6 MET Configuration . . . 543
5.1.20.2.7 Python Embedding . . . 547
5.1.20.2.8 Running METplus . . . 550
5.1.20.2.9 Expected Output . . . 551
5.1.20.2.10 Keywords . . . 551
5.1.20.3 PointStat: Using Python Embedding for Point Observations . . . 551
5.1.20.3.1 Scientific Objective . . . 551
5.1.20.3.2 Datasets . . . 551
5.1.20.3.3 METplus Components . . . 552
5.1.20.3.4 METplus Workflow . . . 552
5.1.20.3.5 METplus Configuration . . . 552
5.1.20.3.6 MET Configuration . . . 556
5.1.20.3.7 Running METplus . . . 560
5.1.20.3.8 Expected Output . . . 561
5.1.20.3.9 Keywords . . . 561
5.1.20.4 PointStat: Once Per Field . . . 561
5.1.20.4.1 Scientific Objective . . . 561
5.1.20.4.2 Datasets . . . 561
5.1.20.4.3 METplus Components . . . 562
5.1.20.4.4 METplus Workflow . . . 562
5.1.20.4.5 METplus Configuration . . . 562
5.1.20.4.6 MET Configuration . . . 566
5.1.20.4.7 Running METplus . . . 571
5.1.20.4.8 Expected Output . . . 571
5.1.20.4.9 Keywords . . . 572
5.1.21 PyEmbedIngest . . . 572
5.1.21.1 PyEmbedIngest: Multiple Fields in One File . . . 572
5.1.21.1.1 Scientific Objective . . . 572
5.1.21.1.2 Datasets . . . 572
5.1.21.1.3 METplus Components . . . 573
5.1.21.1.4 METplus Workflow . . . 573
5.1.21.1.5 METplus Configuration . . . 573
5.1.21.1.6 MET Configuration . . . 575
5.1.21.1.7 Python Embedding . . . 575
5.1.21.1.8 Running METplus . . . 575
5.1.21.1.9 Expected Output . . . 576
5.1.21.1.10 Keywords . . . 576
5.1.21.2 PyEmbedIngest: Basic Use Case . . . 576
5.1.21.2.1 Scientific Objective . . . 577
5.1.21.2.2 Datasets . . . 577
5.1.21.2.3 METplus Components . . . 577
5.1.21.2.4 METplus Workflow . . . 577
5.1.21.2.5 METplus Configuration . . . 577
5.1.21.2.6 MET Configuration . . . 579

5.1.21.2.7 Python Embedding . . . 579
5.1.21.2.8 Running METplus . . . 579
5.1.21.2.9 Expected Output . . . 580
5.1.21.2.10 Keywords . . . 580
5.1.22 RegridDataPlane . . . 581
5.1.22.1 RegridDataPlane: Process all fields . . . 581
5.1.22.1.1 Scientific Objective . . . 581
5.1.22.1.2 Datasets . . . 581
5.1.22.1.3 METplus Components . . . 581
5.1.22.1.4 METplus Workflow . . . 581
5.1.22.1.5 METplus Configuration . . . 582
5.1.22.1.6 MET Configuration . . . 584
5.1.22.1.7 Running METplus . . . 584
5.1.22.1.8 Expected Output . . . 585
5.1.22.1.9 Keywords . . . 585
5.1.22.2 RegridDataPlane: Using Python Embedding . . . 586
5.1.22.2.1 Scientific Objective . . . 586
5.1.22.2.2 Datasets . . . 586
5.1.22.2.3 METplus Components . . . 586
5.1.22.2.4 METplus Workflow . . . 586
5.1.22.2.5 METplus Configuration . . . 587
5.1.22.2.6 MET Configuration . . . 588
5.1.22.2.7 Python Embedding . . . 588
5.1.22.2.8 Running METplus . . . 588
5.1.22.2.9 Expected Output . . . 589
5.1.22.2.10 Keywords . . . 589
5.1.22.3 RegridDataPlane: Run once per field . . . 590
5.1.22.3.1 Scientific Objective . . . 590
5.1.22.3.2 Datasets . . . 590
5.1.22.3.3 METplus Components . . . 590
5.1.22.3.4 METplus Workflow . . . 590
5.1.22.3.5 METplus Configuration . . . 591
5.1.22.3.6 MET Configuration . . . 593
5.1.22.3.7 Running METplus . . . 594
5.1.22.3.8 Expected Output . . . 594
5.1.22.3.9 Keywords . . . 595
5.1.22.4 RegridDataPlane: Basic Use Case . . . 595
5.1.22.4.1 Scientific Objective . . . 595
5.1.22.4.2 Datasets . . . 595
5.1.22.4.3 METplus Components . . . 595
5.1.22.4.4 METplus Workflow . . . 596
5.1.22.4.5 METplus Configuration . . . 596
5.1.22.4.6 MET Configuration . . . 599
5.1.22.4.7 Running METplus . . . 599
5.1.22.4.8 Expected Output . . . 599
5.1.22.4.9 Keywords . . . 600
5.1.23 SeriesAnalysis . . . 600
5.1.23.1 SeriesAnalysis: Using Python Embedding . . . 600

5.1.23.1.1 Scientific Objective . . . 600
5.1.23.1.2 Datasets . . . 600
5.1.23.1.3 METplus Components . . . 601
5.1.23.1.4 METplus Workflow . . . 601
5.1.23.1.5 METplus Configuration . . . 601
5.1.23.1.6 MET Configuration . . . 603
5.1.23.1.7 Python Embedding . . . 606
5.1.23.1.8 Running METplus . . . 606
5.1.23.1.9 Expected Output . . . 607
5.1.23.1.10 Keywords . . . 607
5.1.23.2 SeriesAnalysis: Basic Use Case . . . 608
5.1.23.2.1 Scientific Objective . . . 608
5.1.23.2.2 Datasets . . . 608
5.1.23.2.3 METplus Components . . . 608
5.1.23.2.4 METplus Workflow . . . 608
5.1.23.2.5 METplus Configuration . . . 609
5.1.23.2.6 MET Configuration . . . 612
5.1.23.2.7 Running METplus . . . 615
5.1.23.2.8 Expected Output . . . 616
5.1.23.2.9 Keywords . . . 616
5.1.24 StatAnalysis . . . 617
5.1.24.1 StatAnalysis: Basic Use Case . . . 617
5.1.24.1.1 Scientific Objective . . . 617
5.1.24.1.2 Datasets . . . 617
5.1.24.1.3 METplus Components . . . 617
5.1.24.1.4 METplus Workflow . . . 618
5.1.24.1.5 METplus Configuration . . . 618
5.1.24.1.6 MET Configuration . . . 621
5.1.24.1.7 Running METplus . . . 624
5.1.24.1.8 Expected Output . . . 625
5.1.24.1.9 Keywords . . . 625
5.1.24.2 StatAnalysis: Using Python Embedding . . . 625
5.1.24.2.1 Scientific Objective . . . 625
5.1.24.2.2 Datasets . . . 625
5.1.24.2.3 METplus Components . . . 626
5.1.24.2.4 METplus Workflow . . . 626
5.1.24.2.5 METplus Configuration . . . 626
5.1.24.2.6 MET Configuration . . . 630
5.1.24.2.7 Python Embedding . . . 632
5.1.24.2.8 Running METplus . . . 633
5.1.24.2.9 Expected Output . . . 633
5.1.24.2.10 Keywords . . . 633
5.1.25 TCGen . . . 634
5.1.25.1 TCGen: Basic Use Case . . . 634
5.1.25.1.1 Scientific Objective . . . 634
5.1.25.1.2 Datasets . . . 634
5.1.25.1.3 METplus Components . . . 634
5.1.25.1.4 METplus Workflow . . . 634

5.1.25.1.5 METplus Configuration . . . 635
5.1.25.1.6 MET Configuration . . . 639
5.1.25.1.7 Running METplus . . . 646
5.1.25.1.8 Expected Output . . . 646
5.1.25.1.9 Keywords . . . 647
5.1.26 TCMPRPlotter . . . 647
5.1.26.1 TCMPRPlotter: Basic Use Case . . . 647
5.1.26.1.1 Scientific Objective . . . 647
5.1.26.1.2 Datasets . . . 647
5.1.26.1.3 METplus Components . . . 648
5.1.26.1.4 METplus Workflow . . . 648
5.1.26.1.5 METplus Configuration . . . 648
5.1.26.1.6 MET Configuration . . . 650
5.1.26.1.7 Running METplus . . . 650
5.1.26.1.8 Expected Output . . . 650
5.1.26.1.9 Keywords . . . 651
5.1.27 TCPairs . . . 651
5.1.27.1 TCPairs: Basic Use Case for Extra Tropical Cyclones . . . 651
5.1.27.1.1 Scientific Objective . . . 651
5.1.27.1.2 Datasets . . . 651
5.1.27.1.3 METplus Components . . . 652
5.1.27.1.4 METplus Workflow . . . 652
5.1.27.1.5 METplus Configuration . . . 652
5.1.27.1.6 MET Configuration . . . 654
5.1.27.1.7 Running METplus . . . 658
5.1.27.1.8 Expected Output . . . 659
5.1.27.1.9 Keywords . . . 659
5.1.27.2 TCPairs: Basic Use Case for Tropical Cyclones . . . 659
5.1.27.2.1 Scientific Objective . . . 659
5.1.27.2.2 Datasets . . . 659
5.1.27.2.3 METplus Components . . . 660
5.1.27.2.4 METplus Workflow . . . 660
5.1.27.2.5 METplus Configuration . . . 660
5.1.27.2.6 MET Configuration . . . 662
5.1.27.2.7 Running METplus . . . 666
5.1.27.2.8 Expected Output . . . 667
5.1.27.2.9 Keywords . . . 667
5.1.28 TCRMW . . . 667
5.1.28.1 TCRMW: Basic Use Case . . . 667
5.1.28.1.1 Scientific Objective . . . 667
5.1.28.1.2 Datasets . . . 668
5.1.28.1.3 METplus Components . . . 668
5.1.28.1.4 METplus Workflow . . . 668
5.1.28.1.5 METplus Configuration . . . 668
5.1.28.1.6 MET Configuration . . . 670
5.1.28.1.7 Running METplus . . . 672
5.1.28.1.8 Expected Output . . . 673
5.1.28.1.9 Keywords . . . 673

5.1.29 TCStat 674
5.1.29.1 TCStat: Basic Use Case 674
5.1.29.1.1 Scientific Objective 674
5.1.29.1.2 Datasets 674
5.1.29.1.3 METplus Components 674
5.1.29.1.4 METplus Workflow 674
5.1.29.1.5 METplus Configuration 675
5.1.29.1.6 MET Configuration 678
5.1.29.1.7 Running METplus 682
5.1.29.1.8 Expected Output 683
5.1.29.1.9 Keywords 683
5.1.30 UserScript 684
5.1.30.1 UserScript: Run Once Per Init Use Case 684
5.1.30.1.1 Scientific Objective 684
5.1.30.1.2 Datasets 684
5.1.30.1.3 METplus Components 684
5.1.30.1.4 METplus Workflow 684
5.1.30.1.5 METplus Configuration 684
5.1.30.1.6 MET Configuration 686
5.1.30.1.7 Running METplus 686
5.1.30.1.8 Expected Output 687
5.1.30.1.9 Keywords 688
5.1.30.2 UserScript: Run Once Per Lead Use Case 688
5.1.30.2.1 Scientific Objective 688
5.1.30.2.2 Datasets 688
5.1.30.2.3 METplus Components 689
5.1.30.2.4 METplus Workflow 689
5.1.30.2.5 METplus Configuration 689
5.1.30.2.6 MET Configuration 690
5.1.30.2.7 Running METplus 691
5.1.30.2.8 Expected Output 691
5.1.30.2.9 Keywords 692
5.1.30.3 UserScript: Run Once For Each Runtime Use Case 693
5.1.30.3.1 Scientific Objective 693
5.1.30.3.2 Datasets 693
5.1.30.3.3 METplus Components 693
5.1.30.3.4 METplus Workflow 693
5.1.30.3.5 METplus Configuration 694
5.1.30.3.6 MET Configuration 695
5.1.30.3.7 Running METplus 695
5.1.30.3.8 Expected Output 696
5.1.30.3.9 Keywords 697
5.1.30.4 UserScript: Run Once Per Valid Use Case 697
5.1.30.4.1 Scientific Objective 697
5.1.30.4.2 Datasets 697
5.1.30.4.3 METplus Components 697
5.1.30.4.4 METplus Workflow 697
5.1.30.4.5 METplus Configuration 698

5.1.30.4.6 MET Configuration . . . 699
5.1.30.4.7 Running METplus . . . 699
5.1.30.4.8 Expected Output . . . 700
5.1.30.4.9 Keywords . . . 701
5.1.30.5 UserScript: Run Once Use Case . . . 701
5.1.30.5.1 Scientific Objective . . . 701
5.1.30.5.2 Datasets . . . 702
5.1.30.5.3 METplus Components . . . 702
5.1.30.5.4 METplus Workflow . . . 702
5.1.30.5.5 METplus Configuration . . . 702
5.1.30.5.6 MET Configuration . . . 703
5.1.30.5.7 Running METplus . . . 704
5.1.30.5.8 Expected Output . . . 704
5.1.30.5.9 Keywords . . . 705
5.2 Model Applications . . . 706
5.2.1 Air Quality and Composition . . . 706
5.2.1.1 EnsembleStat: Using Python Embedding for Aerosol Optical Depth . . . 706
5.2.1.1.1 Scientific Objective . . . 706
5.2.1.1.2 Datasets . . . 706
5.2.1.1.3 METplus Components . . . 706
5.2.1.1.4 METplus Workflow . . . 706
5.2.1.1.5 METplus Configuration . . . 707
5.2.1.1.6 MET Configuration . . . 710
5.2.1.1.7 Python Embedding . . . 716
5.2.1.1.8 Running METplus . . . 719
5.2.1.1.9 Expected Output . . . 720
5.2.1.1.10 Keywords . . . 720
5.2.2 Climate . . . 721
5.2.2.1 Grid-Stat: CESM and GFS Analysis CONUS Temp . . . 721
5.2.2.1.1 Scientific Objective . . . 721
5.2.2.1.2 Datasets . . . 721
5.2.2.1.3 METplus Components . . . 721
5.2.2.1.4 METplus Workflow . . . 721
5.2.2.1.5 METplus Configuration . . . 722
5.2.2.1.6 MET Configuration . . . 725
5.2.2.1.7 Running METplus . . . 729
5.2.2.1.8 Expected Output . . . 730
5.2.2.1.9 Keywords . . . 730
5.2.2.2 MODE: CESM and GPCP Asian Monsoon Precipitation . . . 731
5.2.2.2.1 Scientific Objective . . . 731
5.2.2.2.2 Datasets . . . 731
5.2.2.2.3 METplus Components . . . 731
5.2.2.2.4 METplus Workflow . . . 731
5.2.2.2.5 METplus Configuration . . . 732
5.2.2.2.6 MET Configuration . . . 734
5.2.2.2.7 Running METplus . . . 740
5.2.2.2.8 Expected Output . . . 741
5.2.2.2.9 Keywords . . . 742

5.2.3 Convection Allowing Models . . . 742
5.2.3.1 MODE: Brightness Temperature Verification . . . 742
5.2.3.1.1 Scientific Objective . . . 742
5.2.3.1.2 Datasets . . . 742
5.2.3.1.3 METplus Components . . . 743
5.2.3.1.4 METplus Workflow . . . 743
5.2.3.1.5 METplus Configuration . . . 743
5.2.3.1.6 MET Configuration . . . 746
5.2.3.1.7 Running METplus . . . 752
5.2.3.1.8 Expected Output . . . 752
5.2.3.1.9 Keywords . . . 753
5.2.3.2 Grid-Stat: Brightness Temperature Distance Maps . . . 753
5.2.3.2.1 Scientific Objective . . . 753
5.2.3.2.2 Datasets . . . 754
5.2.3.2.3 METplus Components . . . 754
5.2.3.2.4 METplus Workflow . . . 754
5.2.3.2.5 METplus Configuration . . . 754
5.2.3.2.6 MET Configuration . . . 756
5.2.3.2.7 Running METplus . . . 760
5.2.3.2.8 Expected Output . . . 761
5.2.3.2.9 Keywords . . . 762
5.2.3.3 MODE/Grid-Stat: Brightness Temperature Verification and Distance Maps . . . 762
5.2.3.3.1 Scientific Objective . . . 762
5.2.3.3.2 Datasets . . . 762
5.2.3.3.3 METplus Components . . . 762
5.2.3.3.4 METplus Workflow . . . 763
5.2.3.3.5 METplus Configuration . . . 763
5.2.3.3.6 MET Configuration . . . 766
5.2.3.3.7 Running METplus . . . 777
5.2.3.3.8 Expected Output . . . 777
5.2.3.3.9 Keywords . . . 778
5.2.3.4 Point2Grid: Calculate Practically Perfect Probabilities . . . 779
5.2.3.4.1 Scientific Objective . . . 779
5.2.3.4.2 Datasets . . . 779
5.2.3.4.3 METplus Components . . . 779
5.2.3.4.4 METplus Workflow . . . 779
5.2.3.4.5 METplus Configuration . . . 779
5.2.3.4.6 MET Configuration . . . 784
5.2.3.4.7 Python Embedding . . . 785
5.2.3.4.8 Running METplus . . . 787
5.2.3.4.9 Expected Output . . . 788
5.2.3.4.10 Keywords . . . 788
5.2.3.5 Ensemble-Stat: Ensemble Statistics using Obs Uncertainty . . . 789
5.2.3.5.1 Scientific Objective . . . 789
5.2.3.5.2 Datasets . . . 789
5.2.3.5.3 METplus Components . . . 789
5.2.3.5.4 METplus Workflow . . . 789
5.2.3.5.5 METplus Configuration . . . 790

5.2.3.5.6 MET Configuration . . . 795
5.2.3.5.7 Running METplus . . . 801
5.2.3.5.8 Expected Output . . . 802
5.2.3.5.9 Keywords . . . 803
5.2.3.6 METdbLoad: Brightness Temperature . . . 803
5.2.3.6.1 Scientific Objective . . . 803
5.2.3.6.2 Datasets . . . 804
5.2.3.6.3 METplus Components . . . 804
5.2.3.6.4 METplus Workflow . . . 804
5.2.3.6.5 METplus Configuration . . . 804
5.2.3.6.6 XML Configuration . . . 806
5.2.3.6.7 Running METplus . . . 807
5.2.3.6.8 Expected Output . . . 807
5.2.3.6.9 Keywords . . . 808
5.2.3.7 MODE: Hail Verification . . . 808
5.2.3.7.1 Scientific Objective . . . 808
5.2.3.7.2 Datasets . . . 808
5.2.3.7.3 METplus Components . . . 808
5.2.3.7.4 METplus Workflow . . . 809
5.2.3.7.5 METplus Configuration . . . 809
5.2.3.7.6 MET Configuration . . . 811
5.2.3.7.7 Running METplus . . . 817
5.2.3.7.8 Expected Output . . . 818
5.2.3.7.9 Keywords . . . 818
5.2.3.8 Grid-Stat: Surrogate Severe and Practically Perfect Evaluation . . . 819
5.2.3.8.1 Scientific Objective . . . 819
5.2.3.8.2 Datasets . . . 819
5.2.3.8.3 METplus Components . . . 819
5.2.3.8.4 METplus Workflow . . . 819
5.2.3.8.5 METplus Configuration . . . 819
5.2.3.8.6 MET Configuration . . . 822
5.2.3.8.7 Running METplus . . . 827
5.2.3.8.8 Expected Output . . . 827
5.2.3.8.9 Keywords . . . 828
5.2.3.9 Grid-Stat: Surrogate Severe and Practically Perfect Probabilistic Evaluation . . . 828
5.2.3.9.1 Scientific Objective . . . 828
5.2.3.9.2 Datasets . . . 828
5.2.3.9.3 METplus Components . . . 828
5.2.3.9.4 METplus Workflow . . . 829
5.2.3.9.5 METplus Configuration . . . 829
5.2.3.9.6 MET Configuration . . . 832
5.2.3.9.7 Running METplus . . . 836
5.2.3.9.8 Expected Output . . . 837
5.2.3.9.9 Keywords . . . 837
5.2.3.10 Surrogate Severe Calculation: PCPCombine, EnsembleStat, and RegridDataPlane . . . 838
5.2.3.10.1 Scientific Objective . . . 838
5.2.3.10.2 Datasets . . . 838

5.2.3.10.3 METplus Components . . . 838
5.2.3.10.4 METplus Workflow . . . 838
5.2.3.10.5 METplus Configuration . . . 839
5.2.3.10.6 MET Configuration . . . 842
5.2.3.10.7 Running METplus . . . 848
5.2.3.10.8 Expected Output . . . 848
5.2.3.10.9 Keywords . . . 848
5.2.4 Data Assimilation . . . 849
5.2.4.1 StatAnalysis: JEDI . . . 849
5.2.4.1.1 Scientific Objective . . . 849
5.2.4.1.2 Datasets . . . 849
5.2.4.1.3 METplus Components . . . 850
5.2.4.1.4 METplus Workflow . . . 850
5.2.4.1.5 METplus Configuration . . . 850
5.2.4.1.6 MET Configuration . . . 853
5.2.4.1.7 Python Embedding . . . 856
5.2.4.1.8 Running METplus . . . 859
5.2.4.1.9 Expected Output . . . 859
5.2.4.1.10 Keywords . . . 860
5.2.5 Marine and Cryosphere . . . 860
5.2.5.1 GridStat: Python Embedding for sea surface salinity using level 3, 1 day composite obs . . . 860
5.2.5.1.1 Scientific Objective . . . 860
5.2.5.1.2 Datasets . . . 860
5.2.5.1.3 External Dependencies . . . 861
5.2.5.1.4 METplus Components . . . 861
5.2.5.1.5 METplus Workflow . . . 861
5.2.5.1.6 METplus Configuration . . . 862
5.2.5.1.7 MET Configuration . . . 868
5.2.5.1.8 Python Embedding . . . 873
5.2.5.1.9 Running METplus . . . 881
5.2.5.1.10 Expected Output . . . 881
5.2.5.1.11 Keywords . . . 882
5.2.5.2 GridStat: Python Embedding to read and process ice cover . . . 882
5.2.5.2.1 Scientific Objective . . . 882
5.2.5.2.2 Datasets . . . 882
5.2.5.2.3 External Dependencies . . . 883
5.2.5.2.4 METplus Components . . . 883
5.2.5.2.5 METplus Workflow . . . 883
5.2.5.2.6 METplus Configuration . . . 883
5.2.5.2.7 MET Configuration . . . 890
5.2.5.2.8 Python Embedding . . . 894
5.2.5.2.9 Running METplus . . . 899
5.2.5.2.10 Expected Output . . . 900
5.2.5.2.11 Keywords . . . 900
5.2.5.3 Grid-Stat and MODE: Sea Ice Validation . . . 901
5.2.5.3.1 Scientific Objective . . . 901
5.2.5.3.2 Datasets . . . 901

5.2.5.3.3 METplus Components . . . 902
5.2.5.3.4 METplus Workflow . . . 902
5.2.5.3.5 METplus Configuration . . . 902
5.2.5.3.6 MET Configuration . . . 906
5.2.5.3.7 Running METplus . . . 916
5.2.5.3.8 Expected Output . . . 917
5.2.5.3.9 Keywords . . . 918
5.2.5.4 GridStat: Python Embedding for sea surface salinity using level 3, 8 day mean obs . . . 918
5.2.5.4.1 Scientific Objective . . . 918
5.2.5.4.2 Datasets . . . 918
5.2.5.4.3 External Dependencies . . . 919
5.2.5.4.4 METplus Components . . . 919
5.2.5.4.5 METplus Workflow . . . 919
5.2.5.4.6 METplus Configuration . . . 920
5.2.5.4.7 MET Configuration . . . 926
5.2.5.4.8 Python Embedding . . . 931
5.2.5.4.9 Running METplus . . . 939
5.2.5.4.10 Expected Output . . . 939
5.2.5.4.11 Keywords . . . 940
5.2.5.5 GridStat: Python Embedding to read and process SST . . . 940
5.2.5.5.1 Scientific Objective . . . 940
5.2.5.5.2 Datasets . . . 940
5.2.5.5.3 External Dependencies . . . 941
5.2.5.5.4 METplus Components . . . 941
5.2.5.5.5 METplus Workflow . . . 941
5.2.5.5.6 METplus Configuration . . . 941
5.2.5.5.7 MET Configuration . . . 944
5.2.5.5.8 Python Embedding . . . 949
5.2.5.5.9 Running METplus . . . 957
5.2.5.5.10 Expected Output . . . 958
5.2.5.5.11 Keywords . . . 958
5.2.5.6 PlotDataPlane: Python Embedding of tripolar coordinate file . . . 958
5.2.5.6.1 Scientific Objective . . . 959
5.2.5.6.2 Datasets . . . 959
5.2.5.6.3 External Dependencies . . . 959
5.2.5.6.4 METplus Components . . . 959
5.2.5.6.5 METplus Workflow . . . 960
5.2.5.6.6 METplus Configuration . . . 960
5.2.5.6.7 MET Configuration . . . 962
5.2.5.6.8 Python Embedding . . . 962
5.2.5.6.9 Running METplus . . . 966
5.2.5.6.10 Expected Output . . . 967
5.2.5.6.11 Keywords . . . 967
5.2.5.7 GridStat: Python Embedding to read and process sea surface heights . . . 968
5.2.5.7.1 Scientific Objective . . . 968
5.2.5.7.2 Datasets . . . 968
5.2.5.7.3 External Dependencies . . . 968

5.2.5.7.4 METplus Components 969
5.2.5.7.5 METplus Workflow 969
5.2.5.7.6 METplus Configuration 969
5.2.5.7.7 MET Configuration 974
5.2.5.7.8 Python Embedding 979
5.2.5.7.9 Running METplus 987
5.2.5.7.10 Expected Output 988
5.2.5.7.11 Keywords 988
5.2.5.8 UserScript: Python Script to compute cable transport 989
5.2.5.8.1 Scientific Objective 989
5.2.5.8.2 Datasets 989
5.2.5.8.3 External Dependencies 990
5.2.5.8.4 METplus Components 990
5.2.5.8.5 METplus Workflow 990
5.2.5.8.6 METplus Configuration 990
5.2.5.8.7 MET Configuration 992
5.2.5.8.8 User Script 992
5.2.5.8.9 Running METplus 997
5.2.5.8.10 Expected Output 998
5.2.5.8.11 Keywords 998
5.2.6 Medium Range 998
5.2.6.1 Multi_Tool: Feature Relative by Lead using Multiple User-Defined Fields . . . 998
5.2.6.1.1 Scientific Objective 998
5.2.6.1.2 Datasets 999
5.2.6.1.3 External Dependencies 1000
5.2.6.1.4 METplus Components 1000
5.2.6.1.5 METplus Workflow 1000
5.2.6.1.6 METplus Configuration 1001
5.2.6.1.7 MET Configuration 1011
5.2.6.1.8 Python Embedding 1022
5.2.6.1.9 Running METplus 1038
5.2.6.1.10 Expected Output 1039
5.2.6.1.11 Keywords 1040
5.2.6.2 UserScript: Calculate the Difficulty Index 1040
5.2.6.2.1 Scientific Objective 1041
5.2.6.2.2 Datasets 1041
5.2.6.2.3 METplus Components 1041
5.2.6.2.4 METplus Workflow 1041
5.2.6.2.5 METplus Configuration 1042
5.2.6.2.6 MET Configuration 1044
5.2.6.2.7 Python Embedding 1044
5.2.6.2.8 Running METplus 1048
5.2.6.2.9 Expected Output 1049
5.2.6.2.10 Keywords 1050
5.2.6.3 Multi_Tool (MTD): Feature Relative by Lead (with lead groupings) 1050
5.2.6.3.1 Scientific Objective 1050
5.2.6.3.2 Datasets 1050
5.2.6.3.3 METplus Components 1050

5.2.6.3.4 METplus Workflow . . . 1051
5.2.6.3.5 METplus Configuration . . . 1051
5.2.6.3.6 MET Configuration . . . 1054
5.2.6.3.7 Running METplus . . . 1063
5.2.6.3.8 Expected Output . . . 1064
5.2.6.3.9 Keywords . . . 1065
5.2.6.4 Grid-Stat: Standard Verification of Surface Fields . . . 1066
5.2.6.4.1 Scientific Objective . . . 1066
5.2.6.4.2 Datasets . . . 1066
5.2.6.4.3 METplus Components . . . 1066
5.2.6.4.4 METplus Workflow . . . 1066
5.2.6.4.5 METplus Configuration . . . 1067
5.2.6.4.6 MET Configuration . . . 1071
5.2.6.4.7 Running METplus . . . 1075
5.2.6.4.8 Expected Output . . . 1076
5.2.6.4.9 Keywords . . . 1076
5.2.6.5 Point-Stat: Standard Verification for CONUS Surface . . . 1077
5.2.6.5.1 Scientific Objective . . . 1077
5.2.6.5.2 Datasets . . . 1077
5.2.6.5.3 METplus Components . . . 1077
5.2.6.5.4 METplus Workflow . . . 1077
5.2.6.5.5 METplus Configuration . . . 1078
5.2.6.5.6 MET Configuration . . . 1081
5.2.6.5.7 Running METplus . . . 1088
5.2.6.5.8 Expected Output . . . 1089
5.2.6.5.9 Keywords . . . 1089
5.2.6.6 Multi_Tool: Feature Relative by Lead (with lead groupings) . . . 1090
5.2.6.6.1 Scientific Objective . . . 1090
5.2.6.6.2 Datasets . . . 1090
5.2.6.6.3 External Dependencies . . . 1090
5.2.6.6.4 METplus Components . . . 1090
5.2.6.6.5 METplus Workflow . . . 1091
5.2.6.6.6 METplus Configuration . . . 1091
5.2.6.6.7 MET Configuration . . . 1099
5.2.6.6.8 Running METplus . . . 1110
5.2.6.6.9 Expected Output . . . 1111
5.2.6.6.10 Keywords . . . 1112
5.2.6.7 Multi_Tool: Feature Relative by Init . . . 1113
5.2.6.7.1 Scientific Objective . . . 1113
5.2.6.7.2 Datasets . . . 1113
5.2.6.7.3 External Dependencies . . . 1113
5.2.6.7.4 METplus Components . . . 1113
5.2.6.7.5 METplus Workflow . . . 1114
5.2.6.7.6 METplus Configuration . . . 1115
5.2.6.7.7 MET Configuration . . . 1122
5.2.6.7.8 Running METplus . . . 1133
5.2.6.7.9 Expected Output . . . 1134
5.2.6.7.10 Keywords . . . 1135

5.2.6.8 Grid-Stat: Compute Anomaly Correlation using Climatology 1136
5.2.6.8.1 Scientific Objective 1136
5.2.6.8.2 Datasets 1136
5.2.6.8.3 METplus Components 1136
5.2.6.8.4 METplus Workflow 1136
5.2.6.8.5 METplus Configuration 1137
5.2.6.8.6 MET Configuration 1141
5.2.6.8.7 Running METplus 1148
5.2.6.8.8 Expected Output 1149
5.2.6.8.9 Keywords 1149
5.2.6.9 Point-Stat: Standard Verification of Global Upper Air 1150
5.2.6.9.1 Scientific Objective 1150
5.2.6.9.2 Datasets 1150
5.2.6.9.3 METplus Components 1150
5.2.6.9.4 METplus Workflow 1150
5.2.6.9.5 METplus Configuration 1151
5.2.6.9.6 MET Configuration 1154
5.2.6.9.7 Running METplus 1161
5.2.6.9.8 Expected Output 1162
5.2.6.9.9 Keywords 1162
5.2.7 Precipitation 1163
5.2.7.1 Grid-Stat: 6hr QPF in NetCDF format 1163
5.2.7.1.1 Scientific Objective 1163
5.2.7.1.2 Datasets 1163
5.2.7.1.3 METplus Components 1163
5.2.7.1.4 METplus Workflow 1163
5.2.7.1.5 METplus Configuration 1164
5.2.7.1.6 MET Configuration 1166
5.2.7.1.7 Running METplus 1171
5.2.7.1.8 Expected Output 1171
5.2.7.1.9 Keywords 1172
5.2.7.2 Ensemble-Stat: WoFS 1172
5.2.7.2.1 Scientific Objective 1172
5.2.7.2.2 Datasets 1172
5.2.7.2.3 METplus Components 1173
5.2.7.2.4 METplus Workflow 1173
5.2.7.2.5 METplus Configuration 1173
5.2.7.2.6 MET Configuration 1180
5.2.7.2.7 Running METplus 1191
5.2.7.2.8 Expected Output 1191
5.2.7.2.9 Keywords 1192
5.2.7.3 Grid-Stat: 6hr PQPF Probability Verification 1192
5.2.7.3.1 Scientific Objective 1192
5.2.7.3.2 Datasets 1192
5.2.7.3.3 METplus Components 1193
5.2.7.3.4 METplus Workflow 1193
5.2.7.3.5 METplus Configuration 1193
5.2.7.3.6 MET Configuration 1195

5.2.7.3.7 Running METplus 1200
5.2.7.3.8 Expected Output 1201
5.2.7.3.9 Keywords 1201
5.2.7.4 Point-Stat: Investigating Preciptitation Types 1202
5.2.7.4.1 Scientific Objective 1202
5.2.7.4.2 Datasets 1202
5.2.7.4.3 METplus Components 1202
5.2.7.4.4 METplus Workflow 1202
5.2.7.4.5 METplus Configuration 1203
5.2.7.4.6 MET Configuration 1206
5.2.7.4.7 Running METplus 1213
5.2.7.4.8 Expected Output 1214
5.2.7.4.9 Keywords 1214
5.2.7.5 MTD: Build Revision Series to Evaluate Forecast Consistency 1215
5.2.7.5.1 Scientific Objective 1215
5.2.7.5.2 Datasets 1215
5.2.7.5.3 METplus Components 1215
5.2.7.5.4 METplus Workflow 1215
5.2.7.5.5 METplus Configuration 1216
5.2.7.5.6 MET Configuration 1218
5.2.7.5.7 Running METplus 1224
5.2.7.5.8 Expected Output 1225
5.2.7.5.9 Keywords 1225
5.2.7.6 Grid-Stat: 6hr QPF in GEMPAK format 1226
5.2.7.6.1 Scientific Objective 1226
5.2.7.6.2 Datasets 1226
5.2.7.6.3 External Dependencies 1226
5.2.7.6.4 METplus Components 1226
5.2.7.6.5 METplus Workflow 1227
5.2.7.6.6 METplus Configuration 1227
5.2.7.6.7 MET Configuration 1229
5.2.7.6.8 Running METplus 1234
5.2.7.6.9 Expected Output 1235
5.2.7.6.10 Keywords 1235
5.2.7.7 Grid-Stat: 24-hour QPF Use Case 1236
5.2.7.7.1 Scientific Objective 1236
5.2.7.7.2 Datasets 1236
5.2.7.7.3 METplus Components 1236
5.2.7.7.4 METplus Workflow 1236
5.2.7.7.5 METplus Configuration 1237
5.2.7.7.6 MET Configuration 1238
5.2.7.7.7 Running METplus 1243
5.2.7.7.8 Expected Output 1244
5.2.7.7.9 Keywords 1244
5.2.7.8 MTD: 6hr QPF Use Case 1244
5.2.7.8.1 Scientific Objective 1245
5.2.7.8.2 Datasets 1245
5.2.7.8.3 METplus Components 1245

5.2.7.8.4 METplus Workflow . . . 1245
5.2.7.8.5 METplus Configuration . . . 1245
5.2.7.8.6 MET Configuration . . . 1248
5.2.7.8.7 Running METplus . . . 1254
5.2.7.8.8 Expected Output . . . 1254
5.2.7.8.9 Keywords . . . 1255
5.2.7.9 Ensemble-Stat: Basic Post-Processing only . . . 1255
5.2.7.9.1 Scientific Objective . . . 1255
5.2.7.9.2 Datasets . . . 1256
5.2.7.9.3 METplus Components . . . 1256
5.2.7.9.4 METplus Workflow . . . 1256
5.2.7.9.5 METplus Configuration . . . 1257
5.2.7.9.6 MET Configuration . . . 1259
5.2.7.9.7 Running METplus . . . 1265
5.2.7.9.8 Expected Output . . . 1266
5.2.7.9.9 Keywords . . . 1267
5.2.8 Subseasonal to Seasonal . . . 1267
5.2.8.1 TCGen: Genesis Density Function (GDF) and Track Density Function (TDF) . 1267
5.2.8.1.1 Scientific Objective . . . 1267
5.2.8.1.2 Datasets . . . 1269
5.2.8.1.3 Software Versions . . . 1269
5.2.8.1.4 METplus Components . . . 1270
5.2.8.1.5 METplus Workflow . . . 1270
5.2.8.1.6 METplus Configuration . . . 1270
5.2.8.1.7 MET Configuration . . . 1274
5.2.8.1.8 Python Embedding . . . 1281
5.2.8.1.9 Running METplus . . . 1287
5.2.8.1.10 Expected Output . . . 1288
5.2.8.1.11 Keywords . . . 1288
5.2.8.2 UserScript: Make a Cross Spectra plot . . . 1288
5.2.8.2.1 Scientific Objective . . . 1289
5.2.8.2.2 Datasets . . . 1289
5.2.8.2.3 METplus Components . . . 1289
5.2.8.2.4 METplus Workflow . . . 1289
5.2.8.2.5 METplus Configuration . . . 1289
5.2.8.2.6 MET Configuration . . . 1290
5.2.8.2.7 Python Embedding . . . 1290
5.2.8.2.8 Python Scripts . . . 1290
5.2.8.2.9 Running METplus . . . 1293
5.2.8.2.10 Expected Output . . . 1294
5.2.8.2.11 Keywords . . . 1294
5.2.8.3 UserScript: Make a Hovmoeller plot . . . 1294
5.2.8.3.1 Scientific Objective . . . 1295
5.2.8.3.2 Datasets . . . 1295
5.2.8.3.3 METplus Components . . . 1295
5.2.8.3.4 METplus Workflow . . . 1295
5.2.8.3.5 METplus Configuration . . . 1295
5.2.8.3.6 MET Configuration . . . 1296

5.2.8.3.7 Python Embedding 1296
5.2.8.3.8 Running METplus 1296
5.2.8.3.9 Expected Output 1297
5.2.8.3.10 Keywords 1297
5.2.8.4 UserScript: Make zonal and meridonial means 1298
5.2.8.4.1 Scientific Objective 1298
5.2.8.4.2 Datasets 1298
5.2.8.4.3 METplus Components 1298
5.2.8.4.4 METplus Workflow 1298
5.2.8.4.5 METplus Configuration 1298
5.2.8.4.6 MET Configuration 1300
5.2.8.4.7 Python Embedding 1300
5.2.8.4.8 Running METplus 1300
5.2.8.4.9 Expected Output 1301
5.2.8.4.10 Keywords 1301
5.2.8.5 SeriesAnalysis: Standardize ensemble members and calculate probabilistic outputs 1301
5.2.8.5.1 Scientific Objective 1302
5.2.8.5.2 Datasets 1302
5.2.8.5.3 METplus Components 1302
5.2.8.5.4 METplus Workflow 1303
5.2.8.5.5 METplus Configuration 1303
5.2.8.5.6 MET Configuration 1309
5.2.8.5.7 Running METplus 1319
5.2.8.5.8 Expected Output 1320
5.2.8.5.9 Keywords 1320
5.2.8.6 Blocking Calculation: ERA RegridDataPlane, PcpCombine, and Blocking python code 1321
5.2.8.6.1 Scientific Objective 1321
5.2.8.6.2 Datasets 1321
5.2.8.6.3 External Dependencies 1321
5.2.8.6.4 METplus Components 1322
5.2.8.6.5 METplus Workflow 1322
5.2.8.6.6 METplus Configuration 1322
5.2.8.6.7 MET Configuration 1329
5.2.8.6.8 Python Scripts 1329
5.2.8.6.9 Running METplus 1336
5.2.8.6.10 Expected Output 1336
5.2.8.6.11 Keywords 1337
5.2.8.7 GridStat: Determine dominant ensemble members terciles and calculate categorical outputs 1337
5.2.8.7.1 Scientific Objective 1337
5.2.8.7.2 Datasets 1337
5.2.8.7.3 METplus Components 1338
5.2.8.7.4 METplus Workflow 1338
5.2.8.7.5 METplus Configuration 1338
5.2.8.7.6 MET Configuration 1341
5.2.8.7.7 Running METplus 1345

5.2.8.7.8 Expected Output . . . 1346
5.2.8.7.9 Keywords . . . 1346
5.2.8.8 UserScript: Make OMI plot from calculated MJO indices . . . 1347
5.2.8.8.1 Scientific Objective . . . 1347
5.2.8.8.2 Datasets . . . 1347
5.2.8.8.3 External Dependencies . . . 1347
5.2.8.8.4 METplus Components . . . 1348
5.2.8.8.5 METplus Workflow . . . 1348
5.2.8.8.6 METplus Configuration . . . 1348
5.2.8.8.7 MET Configuration . . . 1352
5.2.8.8.8 Python Scripts . . . 1352
5.2.8.8.9 Running METplus . . . 1356
5.2.8.8.10 Expected Output . . . 1357
5.2.8.8.11 Keywords . . . 1357
5.2.8.9 UserScript: Make a Phase Diagram plot from input RMM or OMI . . . 1357
5.2.8.9.1 Scientific Objective . . . 1357
5.2.8.9.2 Datasets . . . 1357
5.2.8.9.3 External Dependencies . . . 1358
5.2.8.9.4 METplus Components . . . 1358
5.2.8.9.5 METplus Workflow . . . 1358
5.2.8.9.6 METplus Configuration . . . 1358
5.2.8.9.7 MET Configuration . . . 1361
5.2.8.9.8 Python Scripts . . . 1361
5.2.8.9.9 Running METplus . . . 1364
5.2.8.9.10 Expected Output . . . 1365
5.2.8.9.11 Keywords . . . 1365
5.2.8.10 WeatherRegime Calculation: ERA RegridDataPlane, PcpCombine, and WeatherRegime python code . . . 1365
5.2.8.10.1 Scientific Objective . . . 1366
5.2.8.10.2 Datasets . . . 1366
5.2.8.10.3 External Dependencies . . . 1366
5.2.8.10.4 METplus Components . . . 1367
5.2.8.10.5 METplus Workflow . . . 1367
5.2.8.10.6 METplus Configuration . . . 1367
5.2.8.10.7 MET Configuration . . . 1373
5.2.8.10.8 Python Scripts . . . 1373
5.2.8.10.9 Running METplus . . . 1379
5.2.8.10.10 Expected Output . . . 1380
5.2.8.10.11 Keywords . . . 1380
5.2.8.11 UserScript: Make RMM plots from calculated MJO indices . . . 1381
5.2.8.11.1 Scientific Objective . . . 1381
5.2.8.11.2 Datasets . . . 1381
5.2.8.11.3 External Dependencies . . . 1381
5.2.8.11.4 METplus Components . . . 1382
5.2.8.11.5 METplus Workflow . . . 1382
5.2.8.11.6 METplus Configuration . . . 1382
5.2.8.11.7 MET Configuration . . . 1393
5.2.8.11.8 Python Scripts . . . 1393

5.2.8.11.9 Running METplus . . . 1401
5.2.8.11.10 Expected Output . . . 1402
5.2.8.11.11 Keywords . . . 1402
5.2.8.12 WeatherRegime Calculation: GFS and ERA RegridDataPlane, PcpCombine, and WeatherRegime python code . . . 1403
5.2.8.12.1 Scientific Objective . . . 1403
5.2.8.12.2 Datasets . . . 1403
5.2.8.12.3 External Dependencies . . . 1403
5.2.8.12.4 METplus Components . . . 1404
5.2.8.12.5 METplus Workflow . . . 1404
5.2.8.12.6 METplus Configuration . . . 1404
5.2.8.12.7 MET Configuration . . . 1413
5.2.8.12.8 Python Scripts . . . 1414
5.2.8.12.9 Running METplus . . . 1420
5.2.8.12.10 Expected Output . . . 1421
5.2.8.12.11 Keywords . . . 1421
5.2.8.13 UserScript: Make OMI plot from calculated MJO indices . . . 1422
5.2.8.13.1 Scientific Objective . . . 1422
5.2.8.13.2 Datasets . . . 1422
5.2.8.13.3 External Dependencies . . . 1422
5.2.8.13.4 METplus Components . . . 1423
5.2.8.13.5 METplus Workflow . . . 1423
5.2.8.13.6 METplus Configuration . . . 1423
5.2.8.13.7 MET Configuration . . . 1429
5.2.8.13.8 Python Scripts . . . 1429
5.2.8.13.9 Running METplus . . . 1433
5.2.8.13.10 Expected Output . . . 1433
5.2.8.13.11 Keywords . . . 1434
5.2.8.14 Grid-Stat and Series-Analysis: BMKG APIK Seasonal Forecast . . . 1434
5.2.8.14.1 Scientific Objective . . . 1434
5.2.8.14.2 Datasets . . . 1436
5.2.8.14.3 METplus Components . . . 1436
5.2.8.14.4 External Dependencies . . . 1436
5.2.8.14.5 METplus Workflow . . . 1436
5.2.8.14.6 METplus Configuration . . . 1437
5.2.8.14.7 MET Configuration . . . 1441
5.2.8.14.8 Running METplus . . . 1448
5.2.8.14.9 Expected Output . . . 1449
5.2.8.14.10 Keywords . . . 1450
5.2.8.15 Blocking Calculation: GFS and ERA RegridDataPlane, PcpCombine, and Blocking python code . . . 1450
5.2.8.15.1 Scientific Objective . . . 1450
5.2.8.15.2 Datasets . . . 1450
5.2.8.15.3 External Dependencies . . . 1451
5.2.8.15.4 METplus Components . . . 1451
5.2.8.15.5 METplus Workflow . . . 1451
5.2.8.15.6 METplus Configuration . . . 1452
5.2.8.15.7 MET Configuration . . . 1465

5.2.8.15.8 Python Scripts 1466
5.2.8.15.9 Running METplus 1472
5.2.8.15.10 Expected Output 1473
5.2.8.15.11 Keywords 1473
5.2.9 Space Weather 1474
5.2.9.1 GenVxMask: Solar Altitude 1474
5.2.9.1.1 Overview 1474
5.2.9.1.2 Scientific Objective 1474
5.2.9.1.3 Datasets 1475
5.2.9.1.4 METplus Use Case Contact 1475
5.2.9.1.5 METplus Components 1475
5.2.9.1.6 METplus Workflow 1475
5.2.9.1.7 METplus Configuration 1476
5.2.9.1.8 MET Configuration 1479
5.2.9.1.9 Running METplus 1479
5.2.9.1.10 Expected Output 1480
5.2.9.1.11 Keywords 1480
5.2.9.2 Grid-Stat: Analysis validation 1480
5.2.9.2.1 Overview 1481
5.2.9.2.2 Scientific Objective 1481
5.2.9.2.3 Datasets 1481
5.2.9.2.4 METplus Use Case Contact 1482
5.2.9.2.5 METplus Components 1482
5.2.9.2.6 METplus Workflow 1482
5.2.9.2.7 METplus Configuration 1482
5.2.9.2.8 MET Configuration 1488
5.2.9.2.9 Running METplus 1492
5.2.9.2.10 Expected Output 1493
5.2.9.2.11 Keywords 1493
5.2.10 Tropical Cyclone and Extra Tropical Cyclone 1494
5.2.10.1 TCRMW: Hurricane Gonzalo 1494
5.2.10.1.1 Scientific Objective 1494
5.2.10.1.2 Datasets 1494
5.2.10.1.3 METplus Components 1494
5.2.10.1.4 METplus Workflow 1494
5.2.10.1.5 METplus Configuration 1495
5.2.10.1.6 MET Configuration 1498
5.2.10.1.7 Running METplus 1500
5.2.10.1.8 Expected Output 1501
5.2.10.1.9 Keywords 1501
5.2.10.2 Track and Intensity Plotter: Generate mean, median and box plots 1502
5.2.10.2.1 Scientific Objective 1502
5.2.10.2.2 Datasets 1502
5.2.10.2.3 METplus Components 1502
5.2.10.2.4 METplus Workflow 1502
5.2.10.2.5 METplus Configuration 1503
5.2.10.2.6 MET Configuration 1507
5.2.10.2.7 Running METplus 1510

5.2.10.2.8 Expected Output . . . 1511
5.2.10.2.9 Keywords . . . 1512
5.2.10.3 CyclonePlotter: Extra-TC Tracker and Plotting Capabilities . . . 1512
5.2.10.3.1 Scientific Objective . . . 1512
5.2.10.3.2 Datasets . . . 1512
5.2.10.3.3 External Dependencies . . . 1513
5.2.10.3.4 METplus Components . . . 1513
5.2.10.3.5 METplus Workflow . . . 1513
5.2.10.3.6 METplus Configuration . . . 1513
5.2.10.3.7 MET Configuration . . . 1517
5.2.10.3.8 Python Embedding . . . 1521
5.2.10.3.9 Running METplus . . . 1525
5.2.10.3.10 Expected Output . . . 1525
5.2.10.3.11 Keywords . . . 1526
5.2.10.4 Grid-Stat: Verification of TC forecasts against merged TDR data . . . 1526
5.2.10.4.1 Scientific Objective . . . 1526
5.2.10.4.2 Datasets . . . 1526
5.2.10.4.3 METplus Components . . . 1527
5.2.10.4.4 METplus Workflow . . . 1527
5.2.10.4.5 METplus Configuration . . . 1527
5.2.10.4.6 MET Configuration . . . 1532
5.2.10.4.7 Python Embedding . . . 1536
5.2.10.4.8 Running METplus . . . 1540
5.2.10.4.9 Expected Output . . . 1541
5.2.10.4.10 Keywords . . . 1541
5.2.10.5 Point-Stat: Standard Verification for CONUS Surface . . . 1541
5.2.10.5.1 Scientific Objective . . . 1541
5.2.10.5.2 Datasets . . . 1542
5.2.10.5.3 METplus Components . . . 1542
5.2.10.5.4 METplus Workflow . . . 1542
5.2.10.5.5 METplus Configuration . . . 1542
5.2.10.5.6 MET Configuration . . . 1545
5.2.10.5.7 Python Embedding . . . 1551
5.2.10.5.8 Running METplus . . . 1555
5.2.10.5.9 Expected Output . . . 1556
5.2.10.5.10 Keywords . . . 1556
5.2.10.6 TCGen: 2021 Global Forecast System (GFS) Tropical Cyclone Genesis Forecast 1557
5.2.10.6.1 Scientific Objective . . . 1557
5.2.10.6.2 Datasets . . . 1557
5.2.10.6.3 METplus Components . . . 1557
5.2.10.6.4 METplus Workflow . . . 1557
5.2.10.6.5 METplus Configuration . . . 1558
5.2.10.6.6 MET Configuration . . . 1562
5.2.10.6.7 Running METplus . . . 1569
5.2.10.6.8 Expected Output . . . 1569
5.2.10.6.9 Keywords . . . 1570
5.2.10.7 Cyclone Plotter: From TC-Pairs Output . . . 1570
5.2.10.7.1 Scientific Objective . . . 1570

5.2.10.7.2 Datasets . . . 1570
5.2.10.7.3 METplus Components . . . 1570
5.2.10.7.4 METplus Workflow . . . 1571
5.2.10.7.5 METplus Configuration . . . 1571
5.2.10.7.6 MET Configuration . . . 1575
5.2.10.7.7 Running METplus . . . 1579
5.2.10.7.8 Expected Output . . . 1580
5.2.10.7.9 Keywords . . . 1580
5.2.10.8 CycloneVerification: TC Verification Compare ADECK vs BDECK . . . 1581
5.2.10.8.1 Scientific Objective . . . 1581
5.2.10.8.2 Datasets . . . 1581
5.2.10.8.3 METplus Workflow . . . 1581
5.2.10.8.4 METplus Components . . . 1582
5.2.10.8.5 METplus Workflow . . . 1582
5.2.10.8.6 METplus Configuration . . . 1582
5.2.10.8.7 MET Configuration . . . 1584
5.2.10.8.8 Running METplus . . . 1592
5.2.10.8.9 Expected Output . . . 1592
5.2.10.8.10Keywords . . . 1593

6 METplus Quick Search for Use Cases 1595
6.1 Use Cases by Application: . . . 1596
6.2 Use Cases by Organization: . . . 1596
6.3 Use Cases by METplus Feature: . . . 1596
6.4 Use cases by File Format: . . . 1597

7 METplus Configuration Glossary 1599

8 METplus Statistics & Diagnostics 1885
8.1 Statistics Database . . . 1886
8.1.1 Statistics List A-B . . . 1886
8.1.2 Statistics List C-E . . . 1888
8.1.3 Statistics List F . . . 1890
8.1.4 Statistics List G-M . . . 1892
8.1.5 Statistics List N-O . . . 1894
8.1.6 Statistics List P-R . . . 1896
8.1.7 Statistics List S-T . . . 1897
8.1.8 Statistics List U-Z . . . 1899
8.2 Diagnostics Database . . . 1900
8.2.1 Diagnostics List A-B . . . 1900
8.2.2 Diagnostics List C-E . . . 1902
8.2.3 Diagnostics List F . . . 1903
8.2.4 Diagnostics List G-L . . . 1904
8.2.5 Diagnostics List M-O . . . 1906
8.2.6 Diagnostics List P-Z . . . 1908

Bibliography 1909

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 development version. 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.

Model Evaluation Tools Plus (METplus) TERMS OF USE - IMPORTANT!

Citations

The citation for this User's Guide should be:

McCabe, G., J. Prestopnik, J. Opatz, J. Halley Gotway, T. Jensen, J. Vigh, M. Row, C. Kalb, H. Fisher, L. Goodrich, D. Adriaansen, M. Win-Gildenmeister, J. Frimel, L. Blank, T. Arbetter, 2022: The METplus Version develop 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. *Software Installation* (page 9) contains basic information about how to get started with METplus Wrappers - including system requirements, required software, and how to download METplus Wrappers. *System Configuration* (page 17) 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 3.6. 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 11) 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 3.6.3+		METplus wrappers, METcalcpy, METplotpy	
Python 3.7		METplus wrappers	
cartopy	>=0.18.0	METplus wrappers, METcalcpy, METplotpy	https://scitools.org.uk/cartopy/docs/late
cfgrib		METplus wrappers	https://pypi.org/project/cfgrib/
cmocean		METcalcpy, METplotpy	https://pypi.org/project/cmocean/
dateutil	>=2.8	METplus wrappers	https://github.com/dateutil/dateutil/rele
eofs		METplus wrappers, METcalcpy, METplotpy	https://pypi.org/project/eofs/

Name	Version	METplus Component	Source
h5py		METplus wrappers	https://github.com/h5py/h5py
imutils	0.5.3	METcalcpy, METplotpy	https://pypi.org/project/imutils/
imageio		METcalcpy, METplotpy	https://pypi.org/project/imageio/
lxml		METcalcpy, METplotpy	https://pypi.org/project/lxml/
matplotlib	>=3.3.4	METplus wrappers, METcalcpy, METplotpy	https://matplotlib.org/stable/users/insta
metcalcpy		METplus wrappers, METcalcpy, METplotpy	https://github.com/dtcenter/METcalcpy/
metplotpy		METplus wrappers	https://github.com/dtcenter/METplotpy/
metpy		METplus wrappers	https://www.unidata.ucar.edu/software/
nc-time-axis	1.4	METplotpy stratosphere_diagnostics	https://github.com/SciTools/nc-time-axi
netCDF4	>=1.5.4	METplus wrappers, METcalcpy, METplotpy	https://unidata.github.io/netcdf4-python
numpy	>=1.19.2	METplus wrappers, METcalcpy, METplotpy	https://numpy.org/
pandas	>=1.0.5	METplus wrappers, METcalcpy, METplotpy	https://pypi.org/project/pandas
pint	>=0.18	METcalcpy	https://github.com/hgrecco/pint
plotly	>=4.9.0	METcalcpy, METplotpy	https://github.com/plotly/plotly.py
psutil	5.7.2	METcalcpy, METplotpy	https://pypi.org/project/psutil/
pygrib		METplus wrappers	https://github.com/jswhit/pygrib
pylab		METplus wrappers	https://pypi.org/project/matplotlib/
pymysql		METcalcpy, METplotpy	https://pypi.org/project/psutil/
pyproj	2.3.1	METplus wrappers	https://github.com/pyproj4/pyproj/arch
pyresample		METplus wrappers	https://github.com/pytroll/pyresample
pytest	>=5.2.1	METcalcpy, METplotpy	https://github.com/pytest-dev/pytest/ar
python-kaleido	>=0.2.1	METcalcpy, METplotpy	https://pypi.org/project/kaleido/
pyyaml	>=5.3.1	METcalcpy, METplotpy	https://github.com/yaml/pyyaml
scikit-image	>=0.16.2	METcalcpy, METplotpy	https://scikit-image.org
scikit-learn	0.23.2	METplus wrappers, METcalcpy, METplotpy	https://github.com/scikit-learn/scikit-lea
scipy	>=1.5.1	METplus wrappers, METcalcpy, METplotpy	https://www.scipy.org/
sklearn		METplus wrappers	https://www.kite.com/python/docs/sklea
statsmodels	>=0.11.1	METcalcpy, METplotpy	https://www.statsmodels.org/
xarray	>=0.17.0	METplus wrappers, METcalcpy, METplotpy	https://xarray.pydata.org/en/v0.17.0/
xesmf		METplus wrappers	NOTE: The xesmf package will not be ins
yaml		METcalcpy, METplotpy	https://pypi.org/project/PyYAML/

1.6 METplus Release Notes

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.

1.6.1 METplus Components Release Note Links

1.6.1.1 Release Notes - Latest Official Release

- MET
- METviewer
- METplotpy
- METcalcpy
- METdatadb
- METexpress
- METplus Wrappers

1.6.1.2 Release Notes - Development Release

- MET
- METviewer
- METplotpy
- METcalcpy
- METdatadb
- METexpress
- METplus Wrappers

1.6.2 METplus Wrappers Release Notes

When applicable, release notes are followed by the GitHub issue number which describes the bugfix, enhancement, or new feature: https://github.com/dtcenter/METplus/issues

1.6.2.1 METplus Version 5.0.0 Beta 1 Release Notes (2022-06-22)

- Enhancements:
  - General:
    - **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)

- Bugfixes:
  - 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: None
- New Use Cases: None
- Documentation:
  - Update documentation to include instructions to disable UserScript wrapper (dtcenter/METplus-Internal#33)
- Internal:
  - 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)

1.7 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.

1.8 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

Software Installation

2.1 Introduction

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

2.2 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.

2.3 Programming/scripting languages

METplus Wrappers is written in Python 3.6.3. 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.

2.4 Requirements

2.4.1 Software Requirements

Minimum Requirements

The following software is required to run METplus Wrappers:

- Python 3.6.3 or above

- MET version 10.0.0 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

2.4.2 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 and the following Python packages installed:

- xarray (0.17.0)
- numpy (1.19.2)
- pandas (1.0.5)
- netCDF4 (1.5.4)

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)

- MakePlots wrapper
    - cartopy (0.18.0)
    - pandas (1.0.5)
- CyclonePlotter wrapper
    - cartopy (0.18.0)
    - matplotlib (3.3.4)

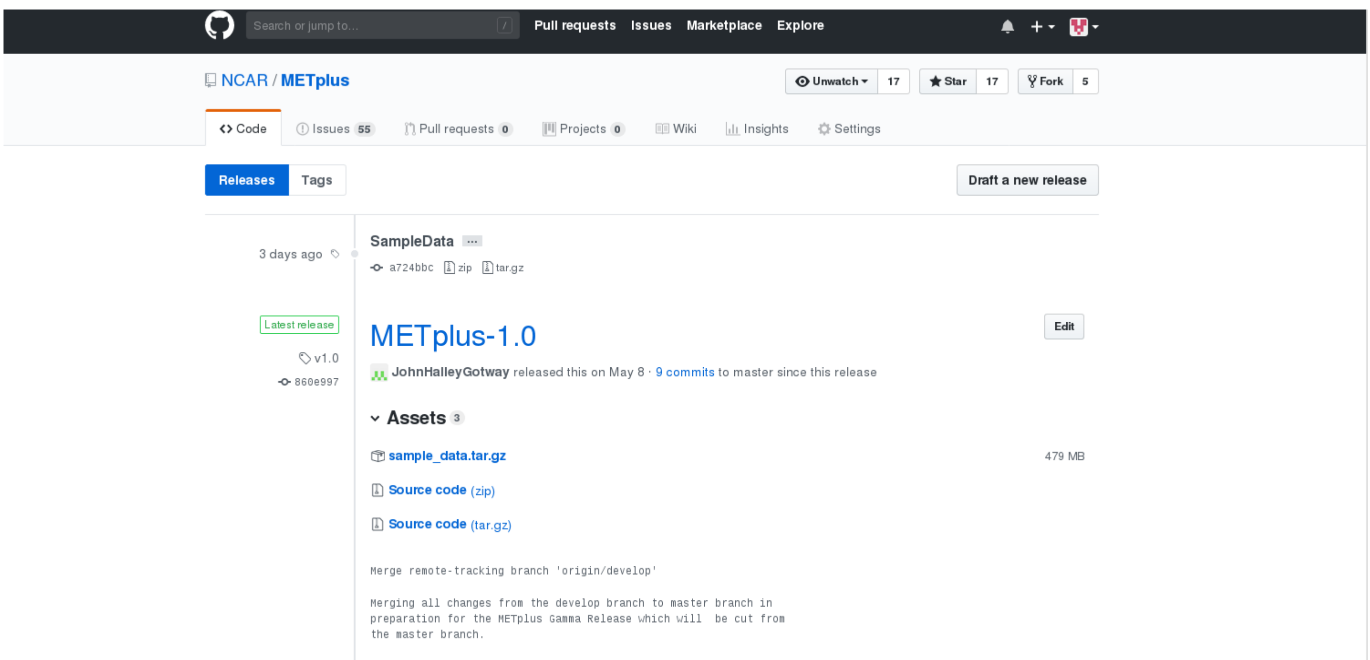
Cartopy, one of the dependencies of CyclonePlotter, attempts to download shapefiles from the internet to complete successfully. So if CyclonePlotter is run on a closed system (i.e. no internet), additional steps need to be taken. First, go to the Natural Earth Data webpage and download the small scale (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.

2.5 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.

2.5.1 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.
- Uncompress the source code (on Linux/Unix*: gunzip* for zip file or *tar xvfz* for the tar.gz file)

2.5.2 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
```

2.6 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.

2.7 METplus Wrappers directory structure

The METplus Wrappers source code contains the following directory structure:

```
METplus/
  build_components/
  docs/
  environment.yml
  internal_tests/
  manage_exernals/
  metplus/
  parm/
  produtil/
  README.md
  requirements.txt
  scripts/
  setup.py
  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_tests/** directory contains test scripts that are only relevant to METplus developers and contributors.

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 **scripts/** directory contains scripts that are used for creating Docker images.

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

2.8 External Components

2.8.1 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

2.9 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.

2.10 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
```

2.11 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 18) in the next chapter.

Chapter 3

System Configuration

This chapter is a guide on configuring METplus Wrappers.

3.1 Config Best Practices / Recommendations

- Set the log level (*LOG_LEVEL* (page 23)) 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.
- 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 25) 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 20)) 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 18).
  - It is recommended to leave **OUTPUT_BASE** set to the default value in the *Default Configuration File* (page 18). 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 18) unchanged and set them in a *User Configuration File* (page 25) 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.

3.2 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 25).

3.2.1 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.

3.2.1.1 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.

3.2.1.2 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
```

3.2.1.3 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
```

3.2.2 Optional

3.2.2.1 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 18) 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
```

3.2.2.2 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 18) 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 that is written in the *OUTPUT_BASE* (page 19) directory:

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

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

3.2.2.3 TMP_DIR

Directory to write any temporary files created by the MET applications. By default, this is a directory inside the *OUTPUT_BASE* (page 19) directory:

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

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

3.2.2.4 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.

By default this is a directory called **stage** inside the *OUTPUT_BASE* (page 19) directory:

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

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

3.2.2.5 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.

3.2.2.6 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.

3.2.2.7 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

3.2.3 Logging

3.2.3.1 Log File Information

Where to write logs files

3.2.3.1.1 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 21) and *LOG_TIMESTAMP_TEMPLATE* (page 22).

3.2.3.1.2 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.

3.2.3.1.3 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 22) 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

3.2.3.1.4 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.

3.2.3.1.5 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.

3.2.3.2 Log Level Information

How much information to log

3.2.3.2.1 LOG_LEVEL

This controls the level of logging output from the METplus wrappers. 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.

3.2.3.2.2 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.

3.2.3.3 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

3.2.3.3.1 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 25).

3.2.3.3.2 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 (met_util.py:218) ERROR: METplus has finished running but had 1␣
↪error.
```

The format of the timestamp is set by *LOG_LINE_DATE_FORMAT* (page 25).

3.2.3.3.3 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 (met_util.py:207) DEBUG: METplus took 0:00:00.850983 to run.
```

The format of the timestamp is set by *LOG_LINE_DATE_FORMAT* (page 25).

3.2.3.3.4 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
```

3.2.3.3.5 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 24) and *LOG_DEBUG_LINE_FORMAT* (page 24).

3.3 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 26) 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 19) 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 18) and/or *INPUT_BASE* (page 19) set in the *default configuration file* (page 18), 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 21) configuration variables are often set in these files, most commonly *LOG_LEVEL* (page 23) = DEBUG to produce additional log output.

3.4 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 28) 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 267) 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 28) section.

3.5 Running METplus

3.5.1 Example Wrapper Use Case

- Create a *User Configuration File* (page 25) (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 (met_util.py:212) 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 25).

This use case does not utilize any of the MET tools, but simply demonstrates how the *Common Config Variables* (page 28) 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 (met_util.py:211) 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 20) file to see all of the settings that were used in the use case.

3.5.2 GridStat Wrapper Basic Use Case

- *Obtain sample input data* (page 12) 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 25) (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 20) file to see all of the settings that were used in the use case.

3.6 Common Config Variables

3.6.1 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.

3.6.1.1 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.

3.6.1.2 Looping by Valid Time

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

3.6.1.2.1 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.

3.6.1.2.2 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.

3.6.1.2.3 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.

3.6.1.2.4 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 33) 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 31) for more information.

3.6.1.2.5 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.

3.6.1.3 Looping by Initialization Time

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

3.6.1.3.1 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.

3.6.1.3.2 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.

3.6.1.3.3 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.

3.6.1.3.4 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 33) 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 31) for more information.

3.6.1.3.5 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.

3.6.1.4 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.

3.6.1.4.1 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 33) 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

You can also define *LEAD_SEQ* using a special notation for many forecast leads. 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 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. You can define any number of these groups by setting configuration variables LEAD_SEQ_1, LEAD_SEQ_2, . . . , *LEAD_SEQ_<n>*. You can define the value with a comma-separated list of integers (currently only hours are supported here) or using the special begin_end_incr(b,e,i) notation described just above. 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
```

3.6.1.4.2 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*.

3.6.1.5 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.

3.6.1.6 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.

You can use **begin_end_incr(b,e,i)** syntax 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"
```

You can also specify multiple strftime directives 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.

3.6.1.7 Realtime Looping

3.6.1.7.1 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. You can also use {today} to substitute the current YYYYMMDD, i.e. 20190426. You cannot change the formatting for the 'today' keyword.

3.6.1.7.2 Shift Keyword

You can use the 'shift' keyword 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 33).

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

3.6.1.7.3 Truncate Keyword

You 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.

3.6.2 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.

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.

3.6.3 Loop Order

The METplus wrappers can be configured to loop first by times then processes or vice-versa. Looping by times first will run each process in the process list for a given run time, increment to the next run time, run each process in the process list, and so on. Looping by processes first will run all times for the first process, then run all times for the second process, and so on.

Example 1 Configuration:

```
[config]
LOOP_ORDER = times

PROCESS_LIST = PCPCombine, GridStat

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

will run in the following order:

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

Example 2 Configuration:

```
[config]
LOOP_ORDER = processes

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
```

Note: If running a MET tool that processes data over a time range, such as SeriesAnalysis or StatAnalysis, the tool must be run with LOOP_ORDER = processes.

3.6.4 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, MakePlots, SeriesByInit, SeriesByLead, 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
```

(continues on next page)

(continued from previous page)

```
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

3.6.5 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.

3.6.5.1 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.

3.6.5.2 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.

3.6.5.3 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.

3.6.5.4 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.

3.6.5.5 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.

3.6.5.6 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}.

3.6.5.7 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.

3.6.5.8 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.

3.6.5.9 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 ]; }];}
```

3.6.5.10 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.

3.6.6 Directory and Filename Template Info

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

3.6.6.1 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.

3.6.6.2 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

3.6.6.3 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

3.6.6.4 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 you want to use the file from the previous day and for the 01Z to 23Z runs you want to 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.

3.6.6.5 Using Windows to find Valid Files

The [FCST/OBS]_FILE_WINDOW_[BEGIN/END] configuration variables can be used if the time information in the input data does not exactly line up with the run time but you still want 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, METplus Wrappers will use /my/grid_stat/input/obs/20190131/pre.20190131_23.ext as the input to grid_stat in this example.

3.6.6.6 Wrapper Specific Windows

A user may need to specify a different window on a wrapper-by-wrapper basis. If this is the case, you 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
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*.

3.6.7 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 you 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 204) : *SERIES_ANALYSIS_RUNTIME_FREQ*
- *GridDiag* (page 127) : *GRID_DIAG_RUNTIME_FREQ*
- *UserScript* (page 265) : *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 *LOOP_ORDER* must be set to processes to run these wrappers. Also 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_ORDER = processes

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2020101712
INIT_END = 2020101912
INIT_INCREMENT = 1d

LEAD_SEQ = 3H, 6H

PROCESS_LIST = SeriesAnalysis

FCST_SERIES_ANALYSIS_INPUT_DIR = /my/fcst/dir
```

(continues on next page)

(continued from previous page)

```
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
```

3.7 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. 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 currently supported by METplus you'd like to control, please refer to: *Overriding Unsupported MET config file settings* (page 68).

The following section demonstrates a few examples using GridStat.

3.7.1 GridStat Simple Example

Visit the *GridStat MET Configuration* (page 135) 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 18). 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.

3.7.2 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.

3.7.3 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 40) section for information on how to sets these values.

3.8 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 18)/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 54) 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.

3.8.1 EnsembleStatConfig

3.8.1.1 message_type

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

3.8.1.2 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

3.8.1.3 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

3.8.1.4 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

3.8.2 GridStatConfig

3.8.2.1 cat_thresh

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

3.8.2.2 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

3.8.2.3 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

3.8.3 MODEConfig

3.8.3.1 grid_res

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

3.8.3.2 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

3.8.3.3 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";}

3.8.3.4 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";}

3.8.3.5 mask_missing_flag

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

3.8.4 PB2NCConfig

3.8.4.1 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

3.8.4.2 quality_mark_thresh

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

3.8.4.3 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

3.8.4.4 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

3.8.5 PointStatConfig

3.8.5.1 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

3.8.5.2 obs_quality

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

3.8.5.3 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

3.8.5.4 interp.type.method and interp.type.width

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

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

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

3.9 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.

3.9.1 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 54) 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.

3.10 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.

You can also reference other variables 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 68).

3.11 Setting Config Variables with Environment Variables

You can set METplus config variables 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.

3.12 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.

3.12.1 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:

```
(met_util.py) ERROR: DEPRECATED CONFIG ITEMS WERE FOUND. PLEASE REMOVE/REPLACE THEM FROM␣
↪CONFIG FILES
(met_util.py) ERROR: [dir] MODEL_DATA_DIR should be replaced with EXTRACT_TILES_GRID_INPUT_
↪DIR
(met_util.py) 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 75).

3.12.2 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:

```
(met_util.py) ERROR: If FCST_VAR1_NAME is set, you must either set OBS_VAR1_NAME or change␣
↪FCST_VAR1_NAME to BOTH_VAR1_NAME
(met_util.py) ERROR: If FCST_VAR2_NAME is set, you must either set OBS_VAR2_NAME or change␣
↪FCST_VAR2_NAME to BOTH_VAR2_NAME
(met_util.py) ERROR: If FCST_VAR1_LEVELS is set, you must either set OBS_VAR1_LEVELS or␣
↪change FCST_VAR1_LEVELS to BOTH_VAR1_LEVELS
(met_util.py) ERROR: If FCST_VAR2_LEVELS is set, you 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 75), but users should review the suggested changes, as they may want to update differently.

3.12.3 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:

```
(met_util.py) 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 75), but users should review the suggested changes, as they may want to include other available input accumulations.

3.12.4 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 75) 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:

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

(met_util.py) ERROR: Please remove deprecated environment variable ${MET_VALID_HHMM} found␣
↪in MET config file: DeprecatedConfig
(met_util.py) ERROR: Set GRID_STAT_CLIMO_MEAN_INPUT_[DIR/TEMPLATE] in a METplus config file␣
↪to set CLIMO_MEAN_FILE in a MET config

(met_util.py) ERROR: output_prefix variable should reference ${OUTPUT_PREFIX} environment␣
↪variable
(met_util.py) INFO: You will need to add GRID_STAT_OUTPUT_PREFIX to the METplus config file␣
↪that sets GRID_STAT_CONFIG_FILE. Set it to:
(met_util.py) 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 75), 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.

3.12.5 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 75) 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    = ${REGRID_TO_GRID};|g' DeprecatedConfig
#Add GRID_STAT_REGRID_TO_GRID
sed -i 's|^    file_name = [ "${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
```

3.12.6 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
```

You must pass a valid configuration to the script, as in you 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 your␣
↪METplus configuration file that sets GRID_STAT_CONFIG_FILE:
```

(continues on next page)

(continued from previous page)

```
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 your 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 4

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.

4.1 ASCII2NC

4.1.1 Description

Used to configure the MET tool ASCII2NC

4.1.2 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

4.1.3 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 54).

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

4.2 CyclonePlotter

4.2.1 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 *Software Installation* (page 9) for details.

4.2.2 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

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

4.3 EnsembleStat

4.3.1 Description

Used to configure the MET tool ensemble_stat.

4.3.2 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
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_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_ENS_VLD_THRESH
ENSEMBLE_STAT_ENS_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_NBRHD_PROB_WIDTH
ENSEMBLE_STAT_NBRHD_PROB_SHAPE
ENSEMBLE_STAT_NBRHD_PROB_VLD_THRESH
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_NMEP_SMOOTH_GAUSSIAN_DX
ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_RADIUS
ENSEMBLE_STAT_NMEP_SMOOTH_VLD_THRESH
ENSEMBLE_STAT_NMEP_SMOOTH_SHAPE
ENSEMBLE_STAT_NMEP_SMOOTH_METHOD
ENSEMBLE_STAT_NMEP_SMOOTH_WIDTH
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_ENSEMBLE_FLAG_LATLON
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN
ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV
ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS
ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS
ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN
ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE
ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT
ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY
ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP

ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK
ENSEMBLE_STAT_ENSEMBLE_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 (optional)
ENS_VAR<n>_NAME (optional)
ENS_VAR<n>_LEVELS (optional)
ENS_VAR<n>_THRESH (optional)
ENS_VAR<n>_OPTIONS (optional)
FCST_ENSEMBLE_STAT_VAR<n>_NAME (optional)
FCST_ENSEMBLE_STAT_VAR<n>_LEVELS (optional)
FCST_ENSEMBLE_STAT_VAR<n>_THRESH (optional)
FCST_ENSEMBLE_STAT_VAR<n>_OPTIONS (optional)
OBS_ENSEMBLE_STAT_VAR<n>_NAME (optional)
OBS_ENSEMBLE_STAT_VAR<n>_LEVELS (optional)
OBS_ENSEMBLE_STAT_VAR<n>_THRESH (optional)
OBS_ENSEMBLE_STAT_VAR<n>_OPTIONS (optional)

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

4.3.3 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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// 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    = "";

//
// Ensemble product fields to be processed
//
ens = {

   ${METPLUS_ENS_FILE_TYPE}

   ${METPLUS_ENS_THRESH}
   ${METPLUS_ENS_VLD_THRESH}
   ${METPLUS_ENS_OBS_THRESH}

   ${METPLUS_ENS_FIELD}
}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}

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

//
// Neighborhood ensemble probabilities
//
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
${METPLUS_NMEP_SMOOTH_DICT}

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

//prob_cat_thresh =
${METPLUS_PROB_CAT_THRESH}
```

(continues on next page)

(continued from previous page)

```
//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_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 ];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}
```

(continues on next page)

(continued from previous page)

```
${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}

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

//
// 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
//
${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}

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

(continues on next page)

(continued from previous page)

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

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

//
// Ensemble product output types
//
${METPLUS_ENSEMBLE_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}
```

${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

${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	ens.ens_thresh

${METPLUS_ENS_VLD_THRESH}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_ENS_VLD_THRESH	ens.vld_thresh

${METPLUS_ENS_OBS_THRESH}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_ENS_OBS_THRESH	ens.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 40) section of the User's Guide.

${METPLUS_NBRHD_PROB_DICT}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_NBRHD_PROB_WIDTH	nbrhd_prob.width
ENSEMBLE_STAT_NBRHD_PROB_SHAPE	nbrhd_prob.shape
ENSEMBLE_STAT_NBRHD_PROB_VLD_THRESH	nbrhd_prob.vld_thresh

${METPLUS_NMEP_SMOOTH_DICT}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_NMEP_SMOOTH_VLD_THRESH	nmep_smooth.vld_thresh
ENSEMBLE_STAT_NMEP_SMOOTH_SHAPE	nmep_smooth.shape
ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_DX	nmep_smooth.gaussian_dx
ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_RADIUS	nmep_smooth.gaussian_radius
ENSEMBLE_STAT_NMEP_SMOOTH_METHOD	nmep_smooth.type.method
ENSEMBLE_STAT_NMEP_SMOOTH_WIDTH	nmep_smooth.type.width

${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 40) 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 40) 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_ENSEMBLE_FLAG_DICT}

METplus Config(s)	MET Config File
ENSEMBLE_STAT_ENSEMBLE_FLAG_LATLON	ensemble_flag.latlon
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN	ensemble_flag.mean
ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV	ensemble_flag.stdev
ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS	ensemble_flag.minus
ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS	ensemble_flag.plus
ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN	ensemble_flag.min
ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX	ensemble_flag.max
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE	ensemble_flag.range
ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT	ensemble_flag.vld_count
ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY	ensemble_flag.frequency
ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP	ensemble_flag.nep
ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP	ensemble_flag.nmep
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK	ensemble_flag.rank
ENSEMBLE_STAT_ENSEMBLE_FLAG_WEIGHT	ensemble_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

4.4 Example

4.4.1 Description

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

4.4.2 Configuration

EXAMPLE_INPUT_DIR
EXAMPLE_INPUT_TEMPLATE
EXAMPLE_CUSTOM_LOOP_LIST

4.5 ExtractTiles

4.5.1 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.

4.5.2 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

4.6 GempakToCF

4.6.1 Description

Used to configure the utility GempakToCF.

4.6.2 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

4.7 GenEnsProd

4.7.1 Description

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

4.7.2 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_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_CDF
GEN_ENS_PROD_ENS_MEMBER_IDS
GEN_ENS_PROD_CONTROL_ID
GEN_ENS_PROD_MET_CONFIG_OVERRIDES

4.7.3 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 54).

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

${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 40) 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_CDF	ensemble_flag.climo_cdf

${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

4.8 GenVxMask

4.8.1 Description

Used to configure the MET tool GenVxMask to define and generate masking regions.

4.8.2 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

4.9 GFDLTracker

4.9.1 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 14) for more information.

4.9.2 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

4.9.3 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

4.10 GridDiag

4.10.1 Description

Used to configure the MET tool grid_diag.

4.10.2 METplus Configuration

GRID_DIAG_INPUT_DIR
GRID_DIAG_OUTPUT_DIR
GRID_DIAG_INPUT_TEMPLATE
GRID_DIAG_OUTPUT_TEMPLATE
GRID_DIAG_VERIFICATION_MASK_TEMPLATE (optional)
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_DESC
GRID_DIAG_SKIP_IF_OUTPUT_EXISTS
GRID_DIAG_RUNTIME_FREQ
GRID_DIAG_DESC
GRID_DIAG_MET_CONFIG_OVERRIDES

4.10.3 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 54).

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

${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 40) 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

4.11 GridStat

4.11.1 Description

Used to configure the MET tool grid_stat.

4.11.2 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 (optional)
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_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_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_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 (optional)
GRID_STAT_MASK_POLY (optional)
GRID_STAT_MET_CONFIG_OVERRIDES
FCST_GRID_STAT_PROB_THRESH (optional)
OBS_GRID_STAT_PROB_THRESH (optional)
GRID_STAT_NEIGHBORHOOD_WIDTH (optional)
GRID_STAT_NEIGHBORHOOD_SHAPE (optional)

GRID_STAT_NEIGHBORHOOD_COV_THRESH (optional)
FCST_GRID_STAT_WINDOW_BEGIN (optional)
FCST_GRID_STAT_WINDOW_END (optional)
OBS_GRID_STAT_WINDOW_BEGIN (optional)
OBS_GRID_STAT_WINDOW_END (optional)
FCST_GRID_STAT_FILE_WINDOW_BEGIN (optional)
FCST_GRID_STAT_FILE_WINDOW_END (optional)
OBS_GRID_STAT_FILE_WINDOW_BEGIN (optional)
OBS_GRID_STAT_FILE_WINDOW_END (optional)
FCST_GRID_STAT_VAR<n>_NAME (optional)
FCST_GRID_STAT_VAR<n>_LEVELS (optional)
FCST_GRID_STAT_VAR<n>_THRESH (optional)
FCST_GRID_STAT_VAR<n>_OPTIONS (optional)
OBS_GRID_STAT_VAR<n>_NAME (optional)
OBS_GRID_STAT_VAR<n>_LEVELS (optional)
OBS_GRID_STAT_VAR<n>_THRESH (optional)
OBS_GRID_STAT_VAR<n>_OPTIONS (optional)

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

4.11.3 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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// 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 =
   ${METPLUS_NBRHD_SHAPE}
```

(continues on next page)

(continued from previous page)

```
   // 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}

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

//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

${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 40) 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 40) 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

${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

${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

4.12 IODA2NC

4.12.1 Description

Used to configure the MET tool ioda2nc

4.12.2 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

4.12.3 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 54).

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

(continues on next page)

(continued from previous page)

```
// 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}

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

//
// Vertical levels to retain
//
// level_range = {
```

(continues on next page)

(continued from previous page)

```
${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
// 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}
```

(continues on next page)

(continued from previous page)

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

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

4.13 MakePlots

4.13.1 Description

The MakePlots wrapper creates various statistical plots using python scripts for the various METplus Wrappers use cases. This can only be run following StatAnalysis wrapper when LOOP_ORDER = processes. To run MakePlots wrapper, include MakePlots in PROCESS_LIST.

4.13.2 METplus Configuration

The following values **must** be defined in the METplus Wrappers configuration file:

MAKE_PLOTS_SCRIPTS_DIR
MAKE_PLOTS_INPUT_DIR
MAKE_PLOTS_OUTPUT_DIR
MAKE_PLOTS_VERIF_CASE
MAKE_PLOTS_VERIF_TYPE
DATE_TYPE
MODEL<n>
MODEL<n>_OBTYPE
MODEL<n>_REFERENCE_NAME
GROUP_LIST_ITEMS
LOOP_LIST_ITEMS

MODEL_LIST
FCST_LEAD_LIST
VX_MASK_LIST
LINE_TYPE_LIST
MAKE_PLOTS_AVERAGE_METHOD
MAKE_PLOTS_STATS_LIST
MAKE_PLOTS_CI_METHOD
MAKE_PLOTS_VERIF_GRID
MAKE_PLOTS_EVENT_EQUALIZATION

The following values are **optional** in the METplus Wrappers configuration file:

VAR<n>_FOURIER_DECOMP
VAR<n>_WAVE_NUM_LIST
FCST_VALID_HOUR_LIST
OBS_VALID_HOUR_LIST
FCST_INIT_HOUR_LIST
OBS_INIT_HOUR_LIST
OBS_LEAD_LIST
DESC_LIST
INTERP_MTHD_LIST
INTERP_PNTS_LIST
COV_THRESH_LIST
ALPHA_LIST

Warning: DEPRECATED:

PLOTTING_SCRIPTS_DIR
STAT_FILES_INPUT_DIR
PLOTTING_OUTPUT_DIR
VERIF_CASE
VERIF_TYPE
PLOT_TIME
MODEL<n>_NAME
MODEL<n>_OBS_NAME

MODEL<n>_NAME_ON_PLOT
VALID_HOUR_METHOD
VALID_HOUR_BEG
VALID_HOUR_END
VALID_HOUR_INCREMENT
INIT_HOUR_BEG
INIT_HOUR_END
INIT_HOUR_INCREMENT
REGION_LIST
LEAD_LIST
LINE_TYPE
INTERP
PLOT_STATS_LIST
CI_METHOD
VERIF_GRID
EVENT_EQUALIZATION

4.14 METdbLoad

4.14.1 Description

Used to call the met_db_load.py script from dtcenter/METdatadb to load MET output into a METviewer database.

4.14.2 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

4.14.3 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>

4.15 MODE

4.15.1 Description

Used to configure the MET Method for Object-based Diagnostic Evaluation tool mode.

4.15.2 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_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
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_MASK_GRID
MODE_MASK_GRID_FLAG
MODE_MASK_POLY
MODE_MASK_POLY_FLAG
MODE_FCST_FILTER_ATTR_NAME
MODE_FCST_FILTER_ATTR_THRESH
MODE_FCST_CENSOR_THRESH
MODE_FCST_CENSOR_VAL
MODE_FCST_VLD_THRESH
MODE_OBS_FILTER_ATTR_NAME
MODE_OBS_FILTER_ATTR_THRESH
MODE_OBS_CENSOR_THRESH
MODE_OBS_CENSOR_VAL
MODE_OBS_VLD_THRESH
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_MASK_MISSING_FLAG
MODE_MATCH_FLAG
MODE_MAX_CENTROID_DIST
MODE_TOTAL_INTEREST_THRESH
MODE_INTEREST_FUNCTION_CENTROID_DIST
MODE_INTEREST_FUNCTION_BOUNDARY_DIST
MODE_INTEREST_FUNCTION_CONVEX_HULL_DIST
MODE_PS_PLOT_FLAG
MODE_CT_STATS_FLAG
FCST_MODE_IS_PROB
FCST_MODE_PROB_IN_GRIB_PDS
MODE_MULTIVAR_LOGIC
FCST_MODE_VAR<n>_NAME (optional)
FCST_MODE_VAR<n>_LEVELS (optional)
FCST_MODE_VAR<n>_THRESH (optional)
FCST_MODE_VAR<n>_OPTIONS (optional)
MODE_FCST_FILE_TYPE (optional)

OBS_MODE_VAR<n>_NAME (optional)
OBS_MODE_VAR<n>_LEVELS (optional)
OBS_MODE_VAR<n>_THRESH (optional)
OBS_MODE_VAR<n>_OPTIONS (optional)
MODE_OBS_FILE_TYPE (optional)

> **Warning: DEPRECATED:**
>
> *MODE_OUT_DIR*
> *MODE_CONFIG*

4.15.3 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/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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// 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}

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FIELD}
```

(continues on next page)

(continued from previous page)

```
   ${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}
}

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}
}

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

//
// 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}
```

${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

${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_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 40) 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_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 40) 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_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

4.16 MTD

4.16.1 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.

4.16.2 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_MET_CONFIG_OVERRIDES
FCST_MTD_IS_PROB
FCST_MTD_PROB_IN_GRIB_PDS
FCST_MTD_VAR<n>_NAME (optional)
FCST_MTD_VAR<n>_LEVELS (optional)
FCST_MTD_VAR<n>_THRESH (optional)

FCST_MTD_VAR<n>_OPTIONS (optional)
OBS_MTD_VAR<n>_NAME (optional)
OBS_MTD_VAR<n>_LEVELS (optional)
OBS_MTD_VAR<n>_THRESH (optional)
OBS_MTD_VAR<n>_OPTIONS (optional)

Warning: DEPRECATED:

MTD_OUT_DIR
MTD_CONFIG
MTD_SINGLE_RUN_SRC

4.16.3 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/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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE Time Domain configuration file.
//
// For additional information, see the MET_BASE/config/README file.
//
////////////////////////////////////////////////////////////////////////////////

   //
   // Output model name to be written
   //

${METPLUS_MODEL}

   //
```

(continues on next page)

(continued from previous page)

```
   // 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;

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

   //
   // 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;

```

(continues on next page)

(continued from previous page)

```
   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 = (

      ( -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 )
```

(continues on next page)

(continued from previous page)

```
      (   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 )

   );

   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 )
```

(continues on next page)

(continued from previous page)

```
      (  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;
   cluster_id   = true;

}

txt_output = {

   attributes_2d   = true;
   attributes_3d   = true;

}


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

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

(continues on next page)

(continued from previous page)

```
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

${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 40) 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 40) 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

4.17 PB2NC

4.17.1 Description

The PB2NC wrapper is a Python script that encapsulates the behavior of the MET pb2nc tool to convert prepBUFR files into netCDF.

4.17.2 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 (optional)
PB2NC_STATION_ID (optional)
PB2NC_GRID (optional)
PB2NC_POLY
PB2NC_OBS_BUFR_VAR_LIST (optional)
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

4.17.3 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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// 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.
//
${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}
```

${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

4.18 PCPCombine

4.18.1 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.

4.18.2 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 (optional)
FCST_PCP_COMBINE_EXTRA_LEVELS (optional)
FCST_PCP_COMBINE_EXTRA_OUTPUT_NAMES (optional)
OBS_PCP_COMBINE_EXTRA_NAMES (optional)
OBS_PCP_COMBINE_EXTRA_LEVELS (optional)
OBS_PCP_COMBINE_EXTRA_OUTPUT_NAMES (optional)
FCST_PCP_COMBINE_OUTPUT_ACCUM (optional)

FCST_PCP_COMBINE_OUTPUT_NAME (optional)
OBS_PCP_COMBINE_OUTPUT_ACCUM (optional)
OBS_PCP_COMBINE_OUTPUT_NAME (optional)

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

4.19 PlotDataPlane

4.19.1 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.

4.19.2 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

4.20 Point2Grid

4.20.1 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

4.20.2 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

4.21 PointStat

4.21.1 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.

4.21.2 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 (optional)
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_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_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 (optional)

FCST_POINT_STAT_WINDOW_END (optional)
OBS_POINT_STAT_WINDOW_BEGIN (optional)
OBS_POINT_STAT_WINDOW_END (optional)
POINT_STAT_NEIGHBORHOOD_WIDTH (optional)
POINT_STAT_NEIGHBORHOOD_SHAPE (optional)
FCST_POINT_STAT_VAR<n>_NAME (optional)
FCST_POINT_STAT_VAR<n>_LEVELS (optional)
FCST_POINT_STAT_VAR<n>_THRESH (optional)
FCST_POINT_STAT_VAR<n>_OPTIONS (optional)
OBS_POINT_STAT_VAR<n>_NAME (optional)
OBS_POINT_STAT_VAR<n>_LEVELS (optional)
OBS_POINT_STAT_VAR<n>_THRESH (optional)
OBS_POINT_STAT_VAR<n>_OPTIONS (optional)
POINT_STAT_OBS_VALID_BEG (optional)
POINT_STAT_OBS_VALID_END (optional)

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

4.21.3 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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// 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_GRID}
   ${METPLUS_MASK_POLY}
   ${METPLUS_MASK_SID}
   //llpnt =
```

(continues on next page)

(continued from previous page)

```
   ${METPLUS_MASK_LLPNT}
}

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

//
// 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}

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

tmp_dir = "${MET_TMP_DIR}";
```

(continues on next page)

(continued from previous page)

```
// 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

${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 40) 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 40) 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_GRID}

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

${METPLUS_MASK_POLY}

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

${METPLUS_MASK_SID}

METplus Config(s)	MET Config File
POINT_STAT_MASK_SID	mask.sid

${METPLUS_MASK_LLPNT}

METplus Config(s)	MET Config File
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_RPS	output_flag.rps
POINT_STAT_OUTPUT_FLAG_ECLV	output_flag.eclv
POINT_STAT_OUTPUT_FLAG_MPR	output_flag.mpr
POINT_STAT_OUTPUT_FLAG_ORANK	output_flag.orank

${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

4.22 PyEmbedIngest

4.22.1 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.

4.22.2 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

4.23 RegridDataPlane

4.23.1 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.

4.23.2 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 (optional)
FCST_REGRID_DATA_PLANE_VAR<n>_INPUT_LEVEL (optional)
FCST_REGRID_DATA_PLANE_VAR<n>_OUTPUT_FIELD_NAME (optional)
OBS_REGRID_DATA_PLANE_VAR<n>_INPUT_FIELD_NAME (optional)
OBS_REGRID_DATA_PLANE_VAR<n>_INPUT_LEVEL (optional)
OBS_REGRID_DATA_PLANE_VAR<n>_OUTPUT_FIELD_NAME (optional)

Warning: DEPRECATED:

VERIFICATION_GRID

4.24 SeriesAnalysis

4.24.1 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*)

4.24.2 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_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

SERIES_ANALYSIS_CLIMO_MEAN_USE_FCST

SERIES_ANALYSIS_CLIMO_MEAN_USE_OBS

SERIES_ANALYSIS_CLIMO_STDEV_FILE_NAME

SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_NAME

SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_LEVELS

SERIES_ANALYSIS_CLIMO_STDEV_VAR<n>_OPTIONS

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

SERIES_ANALYSIS_CLIMO_STDEV_DAY_INTERVAL

SERIES_ANALYSIS_CLIMO_STDEV_HOUR_INTERVAL

SERIES_ANALYSIS_CLIMO_STDEV_FILE_TYPE

SERIES_ANALYSIS_CLIMO_STDEV_USE_FCST

SERIES_ANALYSIS_CLIMO_STDEV_USE_OBS

SERIES_ANALYSIS_CLIMO_CDF_BINS

SERIES_ANALYSIS_CLIMO_CDF_CENTER_BINS

SERIES_ANALYSIS_CLIMO_CDF_DIRECT_PROB

SERIES_ANALYSIS_HSS_EC_VALUE

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

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
FCST_SERIES_ANALYSIS_CAT_THRESH
OBS_SERIES_ANALYSIS_CAT_THRESH
FCST_SERIES_ANALYSIS_IS_PROB
FCST_SERIES_ANALYSIS_PROB_IN_GRIB_PDS

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

4.24.3 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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// 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 = {
   ${METPLUS_OBS_FILE_TYPE}
```

(continues on next page)

(continued from previous page)

```
   ${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 = {
   grid = "";
   poly = "";
}

//
```

(continues on next page)

(continued from previous page)

```
// 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}
```

${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

${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 40) 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 40) 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_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

4.25 SeriesByInit

4.25.1 Description

Warning: This tool has been DEPRECATED. Please use SeriesAnalysis wrapper

4.26 SeriesByLead

4.26.1 Description

Warning: This tool has been DEPRECATED. Please use SeriesAnalysis wrapper

4.27 StatAnalysis

4.27.1 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. StatAnalysis wrapper can be run in two different methods. First is to look at the STAT lines for a single date, to use this method set LOOP_ORDER = times. Second is to look at the STAT lines over a span of dates, to use this method set LOOP_ORDER = processes. To run StatAnalysis wrapper, include StatAnalysis in PROCESS_LIST.

4.27.2 METplus Configuration

The following values must be defined in the METplus Wrappers configuration file for running with LOOP_ORDER = times:

STAT_ANALYSIS_OUTPUT_DIR
MODEL<n>_STAT_ANALYSIS_DUMP_ROW_TEMPLATE
MODEL<n>_STAT_ANALYSIS_OUT_STAT_TEMPLATE
LOG_STAT_ANALYSIS_VERBOSITY
MODEL<n>
MODEL<n>_OBTYPE
MODEL<n>_STAT_ANALYSIS_LOOKIN_DIR
MODEL_LIST
GROUP_LIST_ITEMS
LOOP_LIST_ITEMS
STAT_ANALYSIS_CONFIG_FILE
STAT_ANALYSIS_JOB_NAME
STAT_ANALYSIS_JOB_ARGS
STAT_ANALYSIS_MET_CONFIG_OVERRIDES

The following values are **optional** in the METplus Wrappers configuration file for running with LOOP_ORDER = times:

DESC_LIST
FCST_VALID_HOUR_LIST
OBS_VALID_HOUR_LIST
FCST_INIT_HOUR_LIST
OBS_INIT_HOUR_LIST
FCST_VAR_LIST
OBS_VAR_LIST
FCST_LEVEL_LIST
OBS_LEVEL_LIST
FCST_UNITS_LIST
OBS_UNITS_LIST
FCST_THRESH_LIST
OBS_THRESH_LIST
FCST_LEAD_LIST
OBS_LEAD_LIST
VX_MASK_LIST

INTERP_MTHD_LIST
INTERP_PNTS_LIST
ALPHA_LIST
COV_THRESH_LIST
LINE_TYPE_LIST
STAT_ANALYSIS_SKIP_IF_OUTPUT_EXISTS
STAT_ANALYSIS_HSS_EC_VALUE
STAT_ANALYSIS_OUTPUT_TEMPLATE

The following values **must** be defined in the METplus Wrappers configuration file for running with LOOP_ORDER = processes:

STAT_ANALYSIS_OUTPUT_DIR
LOG_STAT_ANALYSIS_VERBOSITY
DATE_TYPE
STAT_ANALYSIS_CONFIG_FILE
MODEL<n>
MODEL<n>_OBTYPE
MODEL<n>_STAT_ANALYSIS_LOOKIN_DIR
MODEL<n>_REFERENCE_NAME
GROUP_LIST_ITEMS
LOOP_LIST_ITEMS
MODEL_LIST
VX_MASK_LIST
FCST_LEAD_LIST
LINE_TYPE_LIST

The following values are optional in the METplus Wrappers configuration file for running with LOOP_ORDER = processes:

VAR<n>_FOURIER_DECOMP
VAR<n>_WAVE_NUM_LIST
FCST_VALID_HOUR_LIST
OBS_VALID_HOUR_LIST
FCST_INIT_HOUR_LIST
OBS_INIT_HOUR_LIST
OBS_LEAD_LIST

DESC_LIST
INTERP_MTHD_LIST
INTERP_PNTS_LIST
COV_THRESH_LIST
ALPHA_LIST
STAT_ANALYSIS_HSS_EC_VALUE
STAT_ANALYSIS_OUTPUT_TEMPLATE

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

VERIF_CASE
VERIF_TYPE
MODEL<n>_NAME
MODEL<n>_OBS_NAME
MODEL<n>_NAME_ON_PLOT
MODEL<n>_STAT_DIR
REGION_LIST
LEAD_LIST

4.27.3 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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// 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}
```

(continues on next page)

(continued from previous page)

```
${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}

${METPLUS_ALPHA}

${METPLUS_LINE_TYPE}

column = [];

weight = [];

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

//
// Array of STAT-Analysis jobs to be performed on the filtered data
//
```

(continues on next page)

(continued from previous page)

```
${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" ];

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

//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

4.28 TCGen

4.28.1 Description

The TCGen wrapper encapsulates the behavior of the MET tc_gen tool. The wrapper accepts track (Adeck or Bdeck) data and Genesis data.

4.28.2 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

4.28.3 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/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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// 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}

//
// Operational track technique name
//
```

(continues on next page)

(continued from previous page)

```
// 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}

//
// Forecast and operational initialization times to include or exclude
```

(continues on next page)

(continued from previous page)

```
//
// 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
//
// basin_mask =
${METPLUS_BASIN_MASK}
```

(continues on next page)

(continued from previous page)

```
//
// 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
//
// dev_hit_window =
${METPLUS_DEV_HIT_WINDOW_DICT}
```

(continues on next page)

(continued from previous page)

```
// 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}

//
// NetCDF genesis pair counts
//
```

(continues on next page)

(continued from previous page)

```
// 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}
```

${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

4.29 TCMPRPlotter

4.29.1 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.

4.29.2 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

4.30 TCPairs

4.30.1 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.

4.30.2 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_SKIP_LEAD_SEQ
TC_PAIRS_RUN_ONCE
TC_PAIRS_CHECK_DUP
TC_PAIRS_INTERP12

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

4.30.3 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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
// ATCF file format reference:
//   http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html
//

//
// Models
//
${METPLUS_MODEL}

//
// Description
```

(continues on next page)

(continued from previous page)

```
//
${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}

//
// Valid model time window
//
${METPLUS_VALID_BEG}
```

(continues on next page)

(continued from previous page)

```
${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
//
lag_time = [];

//
```

(continues on next page)

(continued from previous page)

```
// 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 = TRUE;

//
// 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;
}

//
// 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_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

${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

4.31 TCRMW

4.31.1 Description

Used to configure the MET tool TC-RMW.

4.31.2 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_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

4.31.3 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/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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// 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.
```

(continues on next page)

(continued from previous page)

```
${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
// 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.
```

(continues on next page)

(continued from previous page)

```
${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}
```

${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 40) 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

${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

4.32 TCStat

4.32.1 Description

Used to configure the MET tool tc_stat.

4.32.2 METplus Configuration

TC_STAT_LOOKIN_DIR
TC_STAT_OUTPUT_DIR
TC_STAT_OUTPUT_TEMPLATE (optional)
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

Warning: DEPRECATED:

TC_STAT_INPUT_DIR
TC_STAT_RUN_VIA
TC_STAT_CMD_LINE_JOB
TC_STAT_JOBS_LIST

4.32.3 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/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 54).

```
////////////////////////////////////////////////////////////////////////////////
//
// 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_INCLUDE}
${METPLUS_INIT_EXCLUDE}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INCLUDE}
${METPLUS_VALID_EXCLUDE}

//
// 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 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}

//
// Stratify by the ADECK and BDECK distances to land.
//
```

(continues on next page)

(continued from previous page)

```
${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}

//
// 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_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_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

4.33 UserScript

4.33.1 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 51) 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.

4.33.2 METplus Configuration

USER_SCRIPT_RUNTIME_FREQ

USER_SCRIPT_COMMAND

USER_SCRIPT_CUSTOM_LOOP_LIST

USER_SCRIPT_SKIP_TIMES

USER_SCRIPT_INPUT_DIR (optional)

USER_SCRIPT_INPUT_TEMPLATE (optional)

USER_SCRIPT_INPUT_TEMPLATE_LABELS (optional)

Chapter 5

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.

5.1 MET tools

5.1.1 ASCII2NC

5.1.1.1 ASCII2NC: Basic Use Case

met_tool_wrapper/ASCII2NC/ASCII2NC.conf

5.1.1.1.1 Scientific Objective

None. Simply converting file formats so point observations can be read by the MET tools.

5.1.1.1.2 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 272) section for more information.

Data Source: Unknown

5.1.1.1.3 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.

5.1.1.1.4 METplus Workflow

ASCII2NC is the only tool called in this example. It processes the following run time:

Valid: 2010-01-01_12Z

5.1.1.1.5 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]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - only ASCII2NC for this case
PROCESS_LIST = ASCII2NC

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 2010010112
```

(continues on next page)

(continued from previous page)

```
# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 2010010112

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 0

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for ASCII2NC only
#LOG_ASCII2NC_VERBOSITY = 1

# MET Configuration file for ASCII2NC
# References CONFIG_DIR from the [dir] section
ASCII2NC_CONFIG_FILE = {CONFIG_DIR}/Ascii2NcConfig_wrapped

# If set to True, skip run if the output file determined by the output directory and
# filename template already exists
ASCII2NC_SKIP_IF_OUTPUT_EXISTS = False

# Time relative to valid time (in seconds if no units are specified) to allow files to be
↪considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
ASCII2NC_FILE_WINDOW_BEGIN = 0
ASCII2NC_FILE_WINDOW_END = 0

# Value to pass with the -format argument to ascii2nc. See MET User's Guide for more
↪information
ASCII2NC_INPUT_FORMAT =

# Value to pass with the -mask_grid argument to ascii2nc. See MET User's Guide for more
↪information
```

(continues on next page)

(continued from previous page)

```
ASCII2NC_MASK_GRID =

# Value to pass with the -mask_poly argument to ascii2nc. See MET User's Guide for more␣
↪information
ASCII2NC_MASK_POLY =

# Value to pass with the -mask_sid argument to ascii2nc. See MET User's Guide for more␣
↪information
ASCII2NC_MASK_SID =

# For defining the time periods for summarization
# False for no time summary, True otherwise
# The rest of the ASCII2NC_TIME_SUMMARY variables are ignored if set to False
# See the MET User's Guide section regarding ASCII2NC time summary options for more␣
↪information.
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


# End of [config] section and start of [dir] section
[dir]
# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/met_config

# Input/Output directories can be left empty if the corresponding template contains the full␣
↪path to the files
ASCII2NC_INPUT_DIR =
ASCII2NC_OUTPUT_DIR =

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for input to ASCII2NC relative to ASCII2NC_INPUT_DIR
ASCII2NC_INPUT_TEMPLATE = {INPUT_BASE}/met_test/data/sample_obs/ascii/precip24_{valid?fmt=%Y
↪%m%d%H}.ascii
```

(continues on next page)

(continued from previous page)

```
# Template to use to write output from ASCII2NC
ASCII2NC_OUTPUT_TEMPLATE = {OUTPUT_BASE}/ascii2nc/precip24_{valid?fmt=%Y%m%d%H}.nc
```

5.1.1.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *ASCII2NC MET Configuration* (page 80) 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}
```

5.1.1.1.7 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.

5.1.1.1.8 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

5.1.1.1.9 Keywords

Note:

- ASCII2NCToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.1.2 ASCII2NC: Using Python Embedding

met_tool_wrapper/ASCII2NC/ASCII2NC_python _embedding.conf

5.1.1.2.1 Scientific Objective

Simply converting file formats so point observations can be read by the MET tools through the use of a Python script

5.1.1.2.2 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 277) section for more information.

Data Source: Unknown

5.1.1.2.3 METplus Components

This use case utilizes the METplus ASCII2NC wrapper to generate a command to run the MET tool ASCII2NC.

5.1.1.2.4 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.

5.1.1.2.5 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]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - only ASCII2NC for this case
PROCESS_LIST = ASCII2NC

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 2010010112

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 2010010112

# Increment between METplus runs (in seconds if no units are specified)
```

(continues on next page)

(continued from previous page)

```
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 0

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for ASCII2NC only
#LOG_ASCII2NC_VERBOSITY = 1

# MET Configuration file for ASCII2NC
# References CONFIG_DIR from the [dir] section
ASCII2NC_CONFIG_FILE =

# If set to True, skip run if the output file determined by the output directory and
# filename template already exists
ASCII2NC_SKIP_IF_OUTPUT_EXISTS = False

# Time relative to valid time (in seconds if no units are specified) to allow files to be␣
↪considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
ASCII2NC_FILE_WINDOW_BEGIN = 0
ASCII2NC_FILE_WINDOW_END = 0

# Time relative to each input file's valid time (in seconds if no units are specified) for␣
↪data within the file to be
# considered valid.
ASCII2NC_WINDOW_BEGIN = 0
ASCII2NC_WINDOW_END = 0

# Value to pass with the -format argument to ascii2nc. See MET User's Guide for more␣
↪information
ASCII2NC_INPUT_FORMAT = python
```

(continues on next page)

(continued from previous page)

```
# Value to pass with the -mask_grid argument to ascii2nc. See MET User's Guide for more␣
↪information
ASCII2NC_MASK_GRID =

# Value to pass with the -mask_poly argument to ascii2nc. See MET User's Guide for more␣
↪information
ASCII2NC_MASK_POLY =

# Value to pass with the -mask_sid argument to ascii2nc. See MET User's Guide for more␣
↪information
ASCII2NC_MASK_SID =

# For defining the time periods for summarization
# False for no time summary, True otherwise
# The rest of the ASCII2NC_TIME_SUMMARY variables are ignored if set to False
# See the MET User's Guide section regarding ASCII2NC time summary options for more␣
↪information.
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


# End of [config] section and start of [dir] section
[dir]
# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/met_config

# Input/Output directories can be left empty if the corresponding template contains the full␣
↪path to the files
ASCII2NC_INPUT_DIR =
ASCII2NC_OUTPUT_DIR =

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for input to ASCII2NC relative to ASCII2NC_INPUT_DIR
ASCII2NC_INPUT_TEMPLATE = "{MET_INSTALL_DIR}/share/met/python/read_ascii_point.py {INPUT_
↪BASE}/met_test/data/sample_obs/ascii/sample_ascii_obs.txt"
```

(continues on next page)

(continued from previous page)

```
# Template to use to write output from ASCII2NC
ASCII2NC_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/ASCII2NC/ascii2nc_python.nc
```

5.1.1.2.6 MET Configuration

None. No MET configuration file for ASCII2NC is used in this case.

5.1.1.2.7 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

5.1.1.2.8 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.

5.1.1.2.9 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

5.1.1.2.10 Keywords

Note:

- ASCII2NCToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.2 Cyclone Plotter

5.1.2.1 CyclonePlotter: Basic Use Case

met_tool_wrapper/CyclonePlotter/CyclonePlotter.conf

5.1.2.1.1 Scientific Objective

Provide visualization of cyclone tracks on a global map (PlateCaree projection)

5.1.2.1.2 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.

5.1.2.1.3 External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- cartopy
- matplotlib

5.1.2.1.4 METplus Components

This use case does not utilize any MET tools

5.1.2.1.5 METplus Workflow

CyclonePlotter is the only tool called in this example. It processes the following run times:

Init: 2015-03-01_12Z

5.1.2.1.6 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

```
[dir]
## Dirs below used by cyclone_plotter_wrapper module.
# --------------------------------------------------
CYCLONE_PLOTTER_INPUT_DIR = {INPUT_BASE}/met_test/tc_pairs
CYCLONE_PLOTTER_OUTPUT_DIR = {OUTPUT_BASE}/cyclone

[config]
# ================================================
```

(continues on next page)

(continued from previous page)

```
PROCESS_LIST = CyclonePlotter

## Config options below used by cyclone_plotter_wrapper module.
# -------------------------------------------------------------
##

#
# Specify the YMD of tracks of interest
#
CYCLONE_PLOTTER_INIT_DATE = 20150301

##
# only 00, 06, 12, and 18z init times are supported in NOAA website,
# so for consistency, these are the only options for METplus.
#
CYCLONE_PLOTTER_INIT_HR = 12 ;; hh format
CYCLONE_PLOTTER_MODEL = GFSO
CYCLONE_PLOTTER_PLOT_TITLE = Model Forecast Storm Tracks
##
# Indicate the region of the globe to plot
#

# Set to Y[y]es or True to plot entire global extent. N[n]o or False
# to generate a plot of a defined region of the world, then define lons and
# lats below.
CYCLONE_PLOTTER_GLOBAL_PLOT = no

##
# Indicate the region (i.e. define a bounding box) to plot
#

# Set to Y[y]es or True to plot entire global extent, N[n]o or False
# to generate a plot of a defined region of the world (and define lons and
# lats below).
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
```

(continues on next page)

(continued from previous page)

```
CYCLONE_PLOTTER_EAST_LON = 179
CYCLONE_PLOTTER_SOUTH_LAT = 0
CYCLONE_PLOTTER_NORTH_LAT = 90


##
#  Indicate the size of symbol (point size)
CYCLONE_PLOTTER_CIRCLE_MARKER_SIZE = 2
CYCLONE_PLOTTER_CROSS_MARKER_SIZE = 11

##
# Indicate text size of annotation label
CYCLONE_PLOTTER_ANNOTATION_FONT_SIZE = 3

##
# Resolution of saved plot in dpi (dots per inch)
# Set to 0 to allow Matplotlib to determine, based on your computer
CYCLONE_PLOTTER_RESOLUTION_DPI = 400


##
#  Turn on/off the generation of an ASCII output file listing all the
#  tracks that are in the plot.  This can be helpful in debugging or verifying
#  that what is plotted is consistent with the data.
#
CYCLONE_PLOTTER_GENERATE_TRACK_ASCII = yes

CYCLONE_PLOTTER_ADD_WATERMARK = False
```

5.1.2.1.7 MET Configuration

No MET configuration is needed to run the cyclone plotter wrapper.

5.1.2.1.8 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.

5.1.2.1.9 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

5.1.2.1.10 Keywords

Note:

- CyclonePlotterUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.3 EnsembleStat

5.1.3.1 EnsembleStat: Using Python Embedding

met_tool_wrapper/EnsembleStat/EnsembleStat_python _embedding.conf

5.1.3.1.1 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.

5.1.3.1.2 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 294) section for more information.

5.1.3.1.3 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.

5.1.3.1.4 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.

5.1.3.1.5 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

```
# Ensemble Stat using Python Embedding Input

[config]

## Configuration-related settings such as the process list, begin and end times, etc.
PROCESS_LIST = EnsembleStat

# Looping by times: steps through each 'task' in the PROCESS_LIST for each
# defined time, and repeats until all times have been evaluated.
LOOP_ORDER = times

# LOOP_BY: Set to INIT to loop over initialization times
LOOP_BY = INIT

# Format of INIT_BEG and INT_END
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run
INIT_BEG=2009123112

# End time for METplus run
INIT_END=2009123112

# Increment between METplus runs in seconds. Must be >= 60
INIT_INCREMENT=3600

# List of forecast leads to process
LEAD_SEQ = 24

# Used in the MET config file for:  model, output_prefix
MODEL = FCST

# Name to identify observation data in output
OBTYPE = OBS

#ENSEMBLE_STAT_DESC =

# The MET ensemble_stat logging level
# 0 quiet to 5 loud, Verbosity setting for MET ensemble_stat output, 2 is default.
# This takes precendence over the general LOG_MET_VERBOSITY set in metplus_logging.conf
#LOG_ENSEMBLE_STAT_VERBOSITY = 2
```

(continues on next page)

(continued from previous page)

```
OBS_ENSEMBLE_STAT_WINDOW_BEGIN = -5400
OBS_ENSEMBLE_STAT_WINDOW_END = 5400

OBS_FILE_WINDOW_BEGIN = 0
OBS_FILE_WINDOW_END = 0

# number of expected members for ensemble. Should correspond with the
# number of items in the list for FCST_ENSEMBLE_STAT_INPUT_TEMPLATE
ENSEMBLE_STAT_N_MEMBERS = 2

# ens.ens_thresh value in the MET config file
# threshold for ratio of valid files to expected files to allow app to run
ENSEMBLE_STAT_ENS_THRESH = 1.0

# ens.vld_thresh value in the MET config file
ENSEMBLE_STAT_ENS_VLD_THRESH = 1.0

# Used in the MET config file for: regrid to_grid field
ENSEMBLE_STAT_REGRID_TO_GRID = NONE

ENSEMBLE_STAT_OUTPUT_PREFIX = PYTHON

ENSEMBLE_STAT_CONFIG_FILE = {PARM_BASE}/met_config/EnsembleStatConfig_wrapped

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_ENSEMBLE_FLAG_LATLON = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP = FALSE
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_WEIGHT = FALSE

# ENSEMBLE_STAT_MET_OBS_ERR_TABLE is not required.
# If the variable is not defined, or the value is not set
# than the MET default is used.
#ENSEMBLE_STAT_MET_OBS_ERR_TABLE =

# Ensemble Variables and levels as specified in the ens field dictionary
# of the MET configuration file. Specify as ENS_VARn_NAME, ENS_VARn_LEVELS,
# (optional) ENS_VARn_OPTION
ENS_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/read_ascii_numpy.py MET_PYTHON_INPUT_ARG␣
↪FCST

# Forecast Variables and levels as specified in the fcst field dictionary
# of the MET configuration file. Specify as FCST_VARn_NAME, FCST_VARn_LEVELS,
# (optional) FCST_VARn_OPTION
FCST_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/read_ascii_numpy.py MET_PYTHON_INPUT_ARG␣
↪FCST

# Observation Variables and levels as specified in the obs field dictionary
# of the MET configuration file. Specify as OBS_VARn_NAME, OBS_VARn_LEVELS,
# (optional) OBS_VARn_OPTION
OBS_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/read_ascii_numpy.py MET_PYTHON_INPUT_ARG␣
↪OBS

ENS_ENSEMBLE_STAT_INPUT_DATATYPE = PYTHON_NUMPY

FCST_ENSEMBLE_STAT_INPUT_DATATYPE = PYTHON_NUMPY

OBS_ENSEMBLE_STAT_INPUT_GRID_DATATYPE = PYTHON_NUMPY

[dir]
# Forecast model input directory for ensemble_stat
FCST_ENSEMBLE_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/python

# Point observation input dir for ensemble_stat
OBS_ENSEMBLE_STAT_POINT_INPUT_DIR =

# Grid observation input dir for ensemble_stat
OBS_ENSEMBLE_STAT_GRID_INPUT_DIR = {INPUT_BASE}/met_test/data/python

# directory containing climatology mean input to EnsembleStat
# Not used in this example
ENSEMBLE_STAT_CLIMO_MEAN_INPUT_DIR =
```

(continues on next page)

(continued from previous page)

```
# directory containing climatology mean input to EnsembleStat
# Not used in this example
ENSEMBLE_STAT_CLIMO_STDEV_INPUT_DIR =

# output directory for ensemble_stat
ENSEMBLE_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/EnsembleStat/ens_python_embedding


[filename_templates]

# FCST_ENSEMBLE_STAT_INPUT_TEMPLATE  - comma separated list of ensemble members
# or a single line, - wildcard characters may be used.

# FCST_ENSEMBLE_STAT_INPUT_TEMPLATE = ????????gep?/d01_{init?fmt=%Y%m%d%H}_02400.grib
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE = fcst.txt, fcst.txt

OBS_ENSEMBLE_STAT_POINT_INPUT_TEMPLATE =

OBS_ENSEMBLE_STAT_GRID_INPUT_TEMPLATE = obs.txt

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

# Template to look for climatology input to EnsembleStat relative to ENSEMBLE_STAT_CLIMO_
↪MEAN_INPUT_DIR
# Not used in this example
ENSEMBLE_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

# Template to look for climatology input to EnsembleStat relative to ENSEMBLE_STAT_CLIMO_
↪STDEV_INPUT_DIR
# Not used in this example
ENSEMBLE_STAT_CLIMO_STDEV_INPUT_TEMPLATE =


ENSEMBLE_STAT_OUTPUT_TEMPLATE =
```

5.1.3.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVER-WRITTEN 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 68)

Note: See the *EnsembleStat MET Configuration* (page 87) 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    = "";

//
// Ensemble product fields to be processed
//
ens = {

   ${METPLUS_ENS_FILE_TYPE}

   ${METPLUS_ENS_THRESH}
   ${METPLUS_ENS_VLD_THRESH}
   ${METPLUS_ENS_OBS_THRESH}

   ${METPLUS_ENS_FIELD}
}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}

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

//
// Neighborhood ensemble probabilities
//
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
${METPLUS_NMEP_SMOOTH_DICT}

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

//prob_cat_thresh =
${METPLUS_PROB_CAT_THRESH}
```

(continues on next page)

(continued from previous page)

```
//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_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 ];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}
```

(continues on next page)

(continued from previous page)

```
${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}

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

//
// 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
//
${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}

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

(continues on next page)

(continued from previous page)

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

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

//
// Ensemble product output types
//
${METPLUS_ENSEMBLE_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}
```

5.1.3.1.7 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

5.1.3.1.8 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.

5.1.3.1.9 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

5.1.3.1.10 Keywords

Note:

- EnsembleStatToolUseCase
- PythonEmbeddingFileUseCase
- EnsembleAppUseCase
- ProbabilityGenerationAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.3.2 EnsembleStat: Basic Use Case

met_tool_wrapper/EnsembleStat/EnsembleStat.conf

5.1.3.2.1 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.

5.1.3.2.2 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 308) section for more information.

Data Source: Unknown

5.1.3.2.3 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.

5.1.3.2.4 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

5.1.3.2.5 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]

PROCESS_LIST = EnsembleStat

###
# Time Info
###
```

(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 = 24H

LOOP_ORDER = times


###
# File I/O
###

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

ENSEMBLE_STAT_N_MEMBERS = 6


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_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
```

(continues on next page)

(continued from previous page)

```
###
# 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}


ENS_VAR1_NAME = APCP
ENS_VAR1_LEVELS = A24
ENS_VAR1_THRESH = >0.0, >=10.0

ENS_VAR2_NAME = REFC
ENS_VAR2_LEVELS = L0
ENS_VAR2_THRESH = >=35.0

ENS_VAR2_OPTIONS = GRIB1_ptv = 129;

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


###
# EnsembleStat
###
```

(continues on next page)

(continued from previous page)

```
#LOG_ENSEMBLE_STAT_VERBOSITY = 2

ENSEMBLE_STAT_CONFIG_FILE = {PARM_BASE}/met_config/EnsembleStatConfig_wrapped

ENSEMBLE_STAT_DESC = NA

OBS_ENSEMBLE_STAT_WINDOW_BEGIN = -5400
OBS_ENSEMBLE_STAT_WINDOW_END = 5400

ENSEMBLE_STAT_ENS_THRESH = 1.0

ENSEMBLE_STAT_ENS_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_CENSOR_THRESH =
ENSEMBLE_STAT_CENSOR_VAL =

ENSEMBLE_STAT_NBRHD_PROB_WIDTH = 5
ENSEMBLE_STAT_NBRHD_PROB_SHAPE = CIRCLE
ENSEMBLE_STAT_NBRHD_PROB_VLD_THRESH = 0.0

ENSEMBLE_STAT_NMEP_SMOOTH_VLD_THRESH = 0.0
ENSEMBLE_STAT_NMEP_SMOOTH_SHAPE = CIRCLE
ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_DX = 81.27
ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_RADIUS = 120
ENSEMBLE_STAT_NMEP_SMOOTH_METHOD = GAUSSIAN
ENSEMBLE_STAT_NMEP_SMOOTH_WIDTH = 1

#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_DUPLICATE_FLAG = NONE
```

(continues on next page)

(continued from previous page)

```
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

#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
```

(continues on next page)

(continued from previous page)

```
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
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_ENSEMBLE_FLAG_LATLON = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK = TRUE
ENSEMBLE_STAT_ENSEMBLE_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 =
```

5.1.3.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *EnsembleStat MET Configuration* (page 87) 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     = "";

//
// Ensemble product fields to be processed
//
ens = {

   ${METPLUS_ENS_FILE_TYPE}

   ${METPLUS_ENS_THRESH}
   ${METPLUS_ENS_VLD_THRESH}
   ${METPLUS_ENS_OBS_THRESH}

   ${METPLUS_ENS_FIELD}
}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}

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

//
// Neighborhood ensemble probabilities
//
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
${METPLUS_NMEP_SMOOTH_DICT}

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

//prob_cat_thresh =
${METPLUS_PROB_CAT_THRESH}
```

(continues on next page)

(continued from previous page)

```
//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_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 ];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}

//obs_quality_exc =
${METPLUS_OBS_QUALITY_EXC}
```

(continues on next page)

(continued from previous page)

```
${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}

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

//
// 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
//
${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}

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

(continues on next page)

(continued from previous page)

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

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

//
// Ensemble product output types
//
${METPLUS_ENSEMBLE_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}
```

5.1.3.2.7 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.

5.1.3.2.8 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

5.1.3.2.9 Keywords

Note:

- EnsembleStatToolUseCase
- PythonEmbeddingFileUseCase
- EnsembleAppUseCase
- ProbabilityGenerationAppUseCase
- GRIBFileUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.4 Example

5.1.4.1 Example: Introductory Use Case

met_tool_wrapper/Example/Example.conf

5.1.4.1.1 Scientific Objective

None.

5.1.4.1.2 Datasets

None.

5.1.4.1.3 METplus Components

This use case utilizes the METplus Example wrapper to demonstrate the effect of time looping and filename template METplus configuration variables.

5.1.4.1.4 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

5.1.4.1.5 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

```
# Example wrapper example

[config]

# List of applications to run - only Example for this case
PROCESS_LIST = Example


# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 2017020100

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 2017020200

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 6H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 3H, 6H, 9H, 12H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
```

(continues on next page)

(continued from previous page)

```
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# list of strings to loop over for each run time.
# value for each item can be referenced in filename templates with {custom?fmt=%s}
EXAMPLE_CUSTOM_LOOP_LIST = ext, nc

# End of [config] section and start of [dir] section
[dir]
# fake directory to look for input data. This can be set to anything, as it only affects the␣
↪log output.
EXAMPLE_INPUT_DIR = /dir/containing/example/data


# End of [dir] section and start of [filename_templates] section
[filename_templates]
# Fake template to use to look for input data. This template is substituted with the time␣
↪information of each
# run time that is executed
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.

5.1.4.1.6 MET Configuration

None.

5.1.4.1.7 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.

5.1.4.1.8 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 (met_util.py:425) INFO: ****************************************
12/30 19:44:02.901 metplus (met_util.py:426) INFO: * Running METplus
12/30 19:44:02.902 metplus (met_util.py:432) INFO: *  at valid time: 201702010000
12/30 19:44:02.902 metplus (met_util.py:435) INFO: ****************************************
12/30 19:44:02.902 metplus.Example (example_wrapper.py:58) INFO: Running ExampleWrapper at␣
↪valid time 20170201000000
12/30 19:44:02.902 metplus.Example (example_wrapper.py:63) INFO: Input directory is /dir/
↪containing/example/data
12/30 19:44:02.902 metplus.Example (example_wrapper.py:64) 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.Example (example_wrapper.py:79) 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.Example (example_wrapper.py:88) INFO: Looking in input directory␣
↪for file: 20170131/file_20170131_21_F003.ext
12/30 19:44:02.903 metplus.Example (example_wrapper.py:79) 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.Example (example_wrapper.py:88) INFO: Looking in input directory␣
↪for file: 20170131/file_20170131_18_F006.ext
12/30 19:44:02.904 metplus.Example (example_wrapper.py:79) 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.Example (example_wrapper.py:88) INFO: Looking in input directory␣
↪for file: 20170131/file_20170131_15_F009.ext
12/30 19:44:02.904 metplus.Example (example_wrapper.py:79) 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.Example (example_wrapper.py:88) INFO: Looking in input directory␣
↪for file: 20170131/file_20170131_12_F012.ext
12/30 19:44:02.904 metplus (met_util.py:425) INFO: ****************************************
12/30 19:44:02.904 metplus (met_util.py:426) INFO: * Running METplus
12/30 19:44:02.905 metplus (met_util.py:432) INFO: *  at valid time: 201702010600
12/30 19:44:02.905 metplus (met_util.py:435) INFO: ****************************************
12/30 19:44:02.905 metplus.Example (example_wrapper.py:58) INFO: Running ExampleWrapper at␣
↪valid time 20170201060000
12/30 19:44:02.905 metplus.Example (example_wrapper.py:63) INFO: Input directory is /dir/
↪containing/example/data
12/30 19:44:02.905 metplus.Example (example_wrapper.py:64) 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.Example (example_wrapper.py:79) INFO: Processing forecast lead 3␣
↪hours initialized at 2017-02-01 03Z and valid at 2017-02-01 06Z
```

(continues on next page)

(continued from previous page)

```
12/30 19:44:02.906 metplus.Example (example_wrapper.py:88) INFO: Looking in input directory␣
↪for file: 20170201/file_20170201_03_F003.ext
```

5.1.4.1.9 Keywords

Note:

- ExampleToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.5 ExtractTiles

5.1.5.1 ExtractTiles: Basic Use Case

met_tool_wrapper/ExtractTiles/ExtractTiles.conf

5.1.5.1.1 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

5.1.5.1.2 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 319) section for more information.

5.1.5.1.3 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.

5.1.5.1.4 METplus Workflow

ExtractTiles is the only tool called in this example. It processes the following run time:

Init: 2014-12-14 0Z

5.1.5.1.5 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]

PROCESS_LIST = ExtractTiles

# The init time begin and end times, increment
LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20141214
INIT_END = 20141214

# Increment in seconds from the begin time to the end time
INIT_INCREMENT = 6H

FCST_VAR1_NAME = TMP
FCST_VAR1_LEVELS = Z2

OBS_VAR1_NAME = TMP
OBS_VAR1_LEVELS = Z2

# Constants used in creating the tile grid
EXTRACT_TILES_NLAT = 60
EXTRACT_TILES_NLON = 60

# Resolution of data in degrees
EXTRACT_TILES_DLAT = 0.5
```

(continues on next page)

(continued from previous page)

```
EXTRACT_TILES_DLON = 0.5

# Degrees to subtract from the center lat and lon to
# calculate the lower left lat (lat_ll) and lower
# left lon (lon_ll) for a grid that is 2n X 2m,
# where n = EXTRACT_TILES_LAT_ADJ degrees and m = EXTRACT_TILES_LON_ADJ degrees.
# For this case, where n=15 and m=15, this results
# in a 30 deg X 30 deg grid
EXTRACT_TILES_LON_ADJ = 15
EXTRACT_TILES_LAT_ADJ = 15

# overwrite modified track data (non-ATCF to ATCF format) if True/yes
EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS = yes

# template of input filter tcst file created by TC-Stat
EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/filter_{init?fmt=%Y%m%d_%H}.tcst

# templates for forecast and observation input 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_TEMPLATE = {valid?fmt=%Y%m%d}/gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}
↪00_000.grb2

# templates for output data
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

# directory containing input filter tcst file created by TC-Stat
EXTRACT_TILES_TC_STAT_INPUT_DIR = {INPUT_BASE}/met_test/extract_tiles

# directory containing gridded input data (forecast and observation)
FCST_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/met_test/new/reduced_model_data
OBS_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/met_test/new/reduced_model_data

# directory to write output
EXTRACT_TILES_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/ExtractTiles
```

5.1.5.1.6 MET Configuration

None. RegridDataPlane does not use configuration files.

5.1.5.1.7 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.

5.1.5.1.8 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

5.1.5.1.9 Keywords

Note:

- RegridDataPlaneToolUseCase
- GRIB2FileUseCase
- FeatureRelativeUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.5.2 ExtractTiles: MTD Input

met_tool_wrapper/ExtractTiles/ExtractTiles_mtd.conf

5.1.5.2.1 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.

5.1.5.2.2 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 327) section for more information.

5.1.5.2.3 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.

5.1.5.2.4 METplus Workflow

ExtractTiles is the only tool called in this example. It processes the following run time:

Init: 2005-08-07 0Z

5.1.5.2.5 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]

PROCESS_LIST = ExtractTiles

# The init time begin and end times, increment
LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2005080700
INIT_END = 2005080700

# Increment in seconds from the begin time to the end time
INIT_INCREMENT = 6H

LEAD_SEQ = 6H

FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A03

OBS_VAR1_NAME = APCP_03
OBS_VAR1_LEVELS = "(*,*)"

# Constants used in creating the tile grid
EXTRACT_TILES_NLAT = 60
EXTRACT_TILES_NLON = 60
```

(continues on next page)

(continued from previous page)

```
# Resolution of data in degrees
EXTRACT_TILES_DLAT = 0.5
EXTRACT_TILES_DLON = 0.5

# Degrees to subtract from the center lat and lon to
# calculate the lower left lat (lat_ll) and lower
# left lon (lon_ll) for a grid that is 2n X 2m,
# where n = EXTRACT_TILES_LAT_ADJ degrees and m = EXTRACT_TILES_LON_ADJ degrees.
# For this case, where n=15 and m=15, this results
# in a 30 deg X 30 deg grid
EXTRACT_TILES_LON_ADJ = 15
EXTRACT_TILES_LAT_ADJ = 15

EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS = no

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
```

5.1.5.2.6 MET Configuration

None. RegridDataPlane does not use configuration files.

5.1.5.2.7 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.

5.1.5.2.8 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

5.1.5.2.9 Keywords

Note:

- RegridDataPlaneToolUseCase
- GRIB2FileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.6 GFDLTracker

5.1.6.1 GFDLTracker: Extra Tropical Cyclone Use Case

met_tool_wrapper/GFDLTracker/GFDLTracker_ETC.conf

5.1.6.1.1 Scientific Objective

Setup and run GFDL Tracker applications to track extra tropical cyclones. See *GFDL Tracker (optional)* (page 14) 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.

5.1.6.1.2 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 335) section for more information.

5.1.6.1.3 METplus Components

This use case utilizes the METplus GFDLTracker wrapper to generate a command to run the GFDL Tracker Fortran applications.

5.1.6.1.4 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)

5.1.6.1.5 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]

PROCESS_LIST = GFDLTracker

LOOP_BY = INIT

INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2021071300
INIT_END = 2021071300
INIT_INCREMENT = 6H

LEAD_SEQ = *

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
```

(continues on next page)

(continued from previous page)

```
GFDL_TRACKER_OUTPUT_TEMPLATE = gfs.{init?fmt=%Y%m%d%H}.etc.txt

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 = "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
```

(continues on next page)

(continued from previous page)

```
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 = ""
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 = ""
```

(continues on next page)

(continued from previous page)

```
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
```

5.1.6.1.6 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},
```

(continues on next page)

(continued from previous page)

```
  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},
  netcdfinfo%z900name = ${METPLUS_NETCDFINFO_Z900NAME},
/
```

(continues on next page)

(continued from previous page)

```
&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},
/
```

5.1.6.1.7 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.

5.1.6.1.8 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

5.1.6.1.9 Keywords

Note:

- GFDLTrackerToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.6.2 GFDLTracker: TC Genesis Use Case

met_tool_wrapper/GFDLTracker/GFDLTracker_Genesis.conf

5.1.6.2.1 Scientific Objective

Setup and run GFDL Tracker applications to track cyclones in TC genesis mode. See *GFDL Tracker (optional)* (page 14) 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.

5.1.6.2.2 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 343) section for more information.

5.1.6.2.3 METplus Components

This use case utilizes the METplus GFDLTracker wrapper to generate a command to run the GFDL Tracker Fortran applications.

5.1.6.2.4 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)

5.1.6.2.5 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]

PROCESS_LIST = GFDLTracker

LOOP_BY = INIT

INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2021071300
INIT_END = 2021071300
INIT_INCREMENT = 6H

LEAD_SEQ = *

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

GFDL_TRACKER_GRIB_VERSION = 2

GFDL_TRACKER_NML_TEMPLATE_FILE = {PARM_BASE}/use_cases/met_tool_wrapper/GFDLTracker/template.
↪nml
```

(continues on next page)

(continued from previous page)

```
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
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"
```

(continues on next page)

(continued from previous page)

```
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 = ""
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
```

(continues on next page)

(continued from previous page)

```
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
```

5.1.6.2.6 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},
```

(continues on next page)

(continued from previous page)

```
  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},
/

&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},
```

(continues on next page)

(continued from previous page)

```
  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},
  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},
```

(continues on next page)

(continued from previous page)

```
  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},
/
```

5.1.6.2.7 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.

5.1.6.2.8 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

5.1.6.2.9 Keywords

Note:

- GFDLTrackerToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.6.3 GFDLTracker: Tropical Cyclone Use Case

met_tool_wrapper/GFDLTracker/GFDLTracker_TC.conf

5.1.6.3.1 Scientific Objective

Setup and run GFDL Tracker applications to track tropical cyclones. See *GFDL Tracker (optional)* (page 14) for more information.

5.1.6.3.2 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 351) section for more information.

5.1.6.3.3 METplus Components

This use case utilizes the METplus GFDLTracker wrapper to generate a command to run the GFDL Tracker Fortran applications.

5.1.6.3.4 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)

5.1.6.3.5 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]

PROCESS_LIST = GFDLTracker

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

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

GFDL_TRACKER_OUTPUT_DIR = {OUTPUT_BASE}/gfdl_tracker/tc
GFDL_TRACKER_OUTPUT_TEMPLATE = hwrf.{init?fmt=%Y%m%d%H}.track.txt

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"
```

(continues on next page)

(continued from previous page)

```
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"
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]"
```

(continues on next page)

(continued from previous page)

```
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_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
```

(continues on next page)

(continued from previous page)

```
GFDL_TRACKER_USER_WANTS_TO_TRACK_ZETA850 = True

GFDL_TRACKER_VERBOSE_VERB = 3
GFDL_TRACKER_VERBOSE_VERB_G2 = 0
```

5.1.6.3.6 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},
```

(continues on next page)

(continued from previous page)

```
  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},
  netcdfinfo%lon_name = ${METPLUS_NETCDFINFO_LON_NAME},
  netcdfinfo%mslpname = ${METPLUS_NETCDFINFO_MSLPNAME},
```

(continues on next page)

(continued from previous page)

```
  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},
  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},
/
```

5.1.6.3.7 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.

5.1.6.3.8 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

5.1.6.3.9 Keywords

Note:

- GFDLTrackerToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.7 GempakToCF

5.1.7.1 GempakToCF: Basic Use Case

met_tool_wrapper/GempakToCF/GempakToCF.conf

5.1.7.1.1 Scientific Objective

None. Simply converting data to a format that MET can read.

5.1.7.1.2 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 355) section for more information.

Data Source: Unknown

5.1.7.1.3 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

5.1.7.1.4 METplus Components

This use case utilizes the METplus GempakToCF wrapper to generate a command to run GempakToCF (external) if all required files are found.

5.1.7.1.5 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

5.1.7.1.6 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

```
# Gempak to NetCDF Configurations

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# List of applications to run - only GempakToCF for this case
PROCESS_LIST = GempakToCF

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
```

(continues on next page)

(continued from previous page)

```
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG=2017062200

# End time for METplus run - must match VALID_TIME_FMT
VALID_END=2017062212

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT=12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# If True, do not run GempakToCF if output file already exists
GEMPAKTOCF_SKIP_IF_OUTPUT_EXISTS = False

[dir]
# input and output data directories
GEMPAKTOCF_INPUT_DIR = {INPUT_BASE}/met_test/new/gempak
GEMPAKTOCF_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/GempakToCF

[filename_templates]
# format of filenames
GEMPAKTOCF_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/mrms_qpe_{valid?fmt=%Y%m%d%H}.grd
GEMPAKTOCF_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/mrms_qpe_{valid?fmt=%Y%m%d%H}.nc
```

5.1.7.1.7 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.

5.1.7.1.8 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

5.1.7.1.9 Keywords

Note:

- GempakToCFToolUseCase
- GEMPAKFileUseCase
- NOAAHMTOrgUseCase
- NOAAWPCOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.8 GenEnsProd

5.1.8.1 GenEnsProd: Basic Use Case

met_tool_wrapper/GenEnsProd/GenEnsProd.conf

5.1.8.1.1 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.

5.1.8.1.2 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 363) section for more information.

5.1.8.1.3 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.

5.1.8.1.4 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

5.1.8.1.5 METplus Configuration

parm/use_cases/met_tool_wrapper/GenEnsProd/GenEnsProd.conf

```
[config]

PROCESS_LIST = GenEnsProd

###
# Time Info
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2009123112
INIT_END=2009123112
INIT_INCREMENT = 12H

LEAD_SEQ = 24H

LOOP_ORDER = processes
```

(continues on next page)

(continued from previous page)

```
###
# File I/O
###

GEN_ENS_PROD_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst

# ensemble gep4 does not exist in sample input data
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

# there are 7 ensembles but 1 is used as control, so specify 6 members
GEN_ENS_PROD_N_MEMBERS = 6

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
###

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

ENS_VAR3_NAME = UGRD
ENS_VAR3_LEVELS = Z10
ENS_VAR3_THRESH = >=5.0
```

(continues on next page)

(continued from previous page)

```
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
###

#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_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
#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 =
```

(continues on next page)

(continued from previous page)

```
#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_CDF = FALSE

#GEN_ENS_PROD_ENS_MEMBER_IDS =
#GEN_ENS_PROD_CONTROL_ID =
```

5.1.8.1.6 MET Configuration

Note: See the *GenEnsProd MET Configuration* (page 104) 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}

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

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

(continues on next page)

(continued from previous page)

```
//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}


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

//
// Neighborhood ensemble probabilities
//
//nbrhd_prob = {
${METPLUS_NBRHD_PROB_DICT}
```

(continues on next page)

(continued from previous page)

```
//
// 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}
```

5.1.8.1.7 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.

5.1.8.1.8 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
```

5.1.8.1.9 Keywords

Note:

- GenEnsProdToolUseCase
- GRIBFileUseCase
- EnsembleAppUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.9 GenVxMask

5.1.9.1 GenVxMask: Multiple Masks

met_tool_wrapper/GenVxMask/GenVxMask_multiple.conf

5.1.9.1.1 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.

5.1.9.1.2 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 368) section for more information.

5.1.9.1.3 METplus Components

This use case utilizes the METplus GenVxMask wrapper to generate a command to run the MET tool GenVx-Mask if all required files are found.

5.1.9.1.4 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.

5.1.9.1.5 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]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - only GenVxMask for this case
PROCESS_LIST = GenVxMask

# time looping - 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
LOOP_BY = INIT

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG = 2005080700

# End time for METplus run - must match INIT_TIME_FMT
INIT_END = 2005080700

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 24H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
```

(continues on next page)

(continued from previous page)

```
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for GenVxMask only
#LOG_GEN_VX_MASK_VERBOSITY = 2

GEN_VX_MASK_SKIP_IF_OUTPUT_EXISTS = False

# Time relative to valid time (in seconds if no units are specified) to allow files to be␣
↪considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
GEN_VX_MASK_FILE_WINDOW_BEGIN = 0
GEN_VX_MASK_FILE_WINDOW_END = 0

GEN_VX_MASK_OPTIONS = -type lat -thresh 'ge30&&le50', -type lon -thresh 'le-70&&ge-130' -
↪intersection -name lat_lon_mask

[filename_templates]

# Template to look for input to GenVxMask relative to 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_TEMPLATE = LATLON_GRID, LATLON_GRID

# Template to use to write output from GenVxMask
GEN_VX_MASK_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/GenVxMask/LAT_LON_mask.nc

[dir]

# Input/Output directories can be left empty if the corresponding template contains the full␣
↪path to the files
GEN_VX_MASK_INPUT_DIR =

GEN_VX_MASK_INPUT_MASK_DIR =

GEN_VX_MASK_OUTPUT_DIR =
```

5.1.9.1.6 MET Configuration

None. GenVxMask does not use configuration files.

5.1.9.1.7 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.

5.1.9.1.8 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

5.1.9.1.9 Keywords

Note:

- GenVxMaskToolUseCase
- GRIBFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.9.2 GenVxMask: Using Arguments

met_tool_wrapper/GenVxMask/GenVxMask_with_arguments.conf

5.1.9.2.1 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.

5.1.9.2.2 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 372) section for more information.

5.1.9.2.3 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.

5.1.9.2.4 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.

5.1.9.2.5 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]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - only GenVxMask for this case
PROCESS_LIST = GenVxMask

# time looping - 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
LOOP_BY = INIT

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG = 2005080700
```

(continues on next page)

(continued from previous page)

```
# End time for METplus run - must match INIT_TIME_FMT
INIT_END = 2005080700

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 24H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for GenVxMask only
#LOG_GEN_VX_MASK_VERBOSITY = 2

GEN_VX_MASK_SKIP_IF_OUTPUT_EXISTS = False

# Time relative to valid time (in seconds if no units are specified) to allow files to be␣
↪considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
GEN_VX_MASK_FILE_WINDOW_BEGIN = 0
GEN_VX_MASK_FILE_WINDOW_END = 0

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"

[filename_templates]

# Template to look for input to GenVxMask relative to 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_TEMPLATE = {GEN_VX_MASK_INPUT_TEMPLATE}
```

(continues on next page)

(continued from previous page)

```
# Template to use to write output from GenVxMask
GEN_VX_MASK_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/GenVxMask/DATA_INPUT_FIELD_APCP_
↪{lead?fmt=%2H}_where_TMP_Z2_le300.nc

[dir]

# Input/Output directories can be left empty if the corresponding template contains the full␣
↪path to the files
GEN_VX_MASK_INPUT_DIR =

GEN_VX_MASK_INPUT_MASK_DIR =

GEN_VX_MASK_OUTPUT_DIR =
```

5.1.9.2.6 MET Configuration

None. GenVxMask does not use configuration files.

5.1.9.2.7 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.

5.1.9.2.8 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

5.1.9.2.9 Keywords

Note:

- GenVxMaskToolUseCase
- GRIBFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.9.3 GenVxMask: Basic Use Case

met_tool_wrapper/GenVxMask/GenVxMask.conf

5.1.9.3.1 Scientific Objective

Creating masking region files to be used by other MET tools.

5.1.9.3.2 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 377) section for more information.

5.1.9.3.3 METplus Components

This use case utilizes the METplus GenVxMask wrapper to generate a command to run the MET tool GenVx-Mask if all required files are found.

5.1.9.3.4 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.

5.1.9.3.5 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]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - only GenVxMask for this case
PROCESS_LIST = GenVxMask

# time looping - 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
LOOP_BY = INIT

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG = 2012040900

# End time for METplus run - must match INIT_TIME_FMT
INIT_END = 2012040900

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 12H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
```

(continues on next page)

(continued from previous page)

```
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for GenVxMask only
#LOG_GEN_VX_MASK_VERBOSITY = 1

GEN_VX_MASK_SKIP_IF_OUTPUT_EXISTS = False

# Time relative to valid time (in seconds if no units are specified) to allow files to be␣
↪considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
GEN_VX_MASK_FILE_WINDOW_BEGIN = 0
GEN_VX_MASK_FILE_WINDOW_END = 0

# Options to add to the gen_vx_mask command line arguments. See MET User's Guide for more␣
↪information
# This can be a comma separated list of options to run GenVxMask multiple times
# The length of this list must be the same length as the GEN_VX_MASK_INPUT_MASK_TEMPLATE list
GEN_VX_MASK_OPTIONS = -type poly

# End of [config] section and start of [filename_templates] section
[filename_templates]

# Template to look for input to GenVxMask relative to 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_TEMPLATE = {INPUT_BASE}/met_test/data/poly/CONUS.poly

# Template to use to write output from GenVxMask
# This can be a comma separated list of options to run GenVxMask multiple times
# The length of this list must be the same length as the GEN_VX_MASK_OPTIONS list
GEN_VX_MASK_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/GenVxMask/POLY_GFS_LATLON_CONUS_
↪mask.nc

# End of [filename_templates] section and start of [dir] section
[dir]

# Input/Output directories can be left empty if the corresponding template contains the full␣
↪path to the files
GEN_VX_MASK_INPUT_DIR =

GEN_VX_MASK_INPUT_MASK_DIR =

GEN_VX_MASK_OUTPUT_DIR =
```

5.1.9.3.6 MET Configuration

None. GenVxMask does not use configuration files.

5.1.9.3.7 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.

5.1.9.3.8 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

5.1.9.3.9 Keywords

Note:

- GenVxMaskToolUseCase
- GRIBFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.10 GridDiag

5.1.10.1 GridDiag: Basic Use Case

met_tool_wrapper/GridDiag/GridDiag.conf

5.1.10.1.1 Scientific Objective

The Grid-Diag tool creates histograms (probability distributions when normalized) for an arbitrary collection of data fields and levels.

5.1.10.1.2 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 382) section for more information.

5.1.10.1.3 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.

5.1.10.1.4 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

5.1.10.1.5 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

```
#
#  CONFIGURATION
#
[config]

# Looping by times: steps through each 'task' in the PROCESS_LIST for each
# defined time, and repeats until all times have been evaluated.
LOOP_ORDER = processes

# 'Tasks' to be run
PROCESS_LIST = GridDiag

LOOP_BY = INIT

# The init time begin and end times, increment, and last init hour.
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2016092900
INIT_END = 2016092900

# This is the step-size. Increment in seconds from the begin time to the end time
# set to 6 hours = 21600 seconds
INIT_INCREMENT = 21600

LEAD_SEQ = 141, 144, 147
```

(continues on next page)

(continued from previous page)

```
# frequency to run the tool
# valid options include:
# RUN_ONCE, RUN_ONCE_PER_INIT_OR_VALID, RUN_ONCE_PER_LEAD, RUN_ONCE_FOR_EACH
GRID_DIAG_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

#LOG_GRID_DIAG_VERBOSITY = 2

GRID_DIAG_DESC = GFS

# Configuration file
GRID_DIAG_CONFIG_FILE = {CONFIG_DIR}/GridDiagConfig_wrapped

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];

# The following variables set values in the MET
# configuration file used by this example
# Leaving these values commented will use the value
# found in the default MET configuration file
#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_MASK_POLY = MET_BASE/poly/SAO.poly

#
#  DIRECTORIES
#
[dir]

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/met_config

GRID_DIAG_INPUT_DIR = {INPUT_BASE}/met_test/new/model_data/grib2/gfs_fv3

GRID_DIAG_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/GridDiag
```

(continues on next page)

(continued from previous page)

```
[filename_templates]

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_TEMPLATE = grid_diag_out.nc
```

5.1.10.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridDiag MET Configuration* (page 128) 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}

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

//
```

(continues on next page)

(continued from previous page)

```
// 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}
```

5.1.10.1.7 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.

5.1.10.1.8 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

5.1.10.1.9 Keywords

Note:

- GridDiagToolUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.11 GridStat

5.1.11.1 GridStat: Basic Use Case

met_tool_wrapper/GridStat/GridStat.conf

5.1.11.1.1 Scientific Objective

Compare 3 hour forecast precipitation accumulations to observations of 3 hour precipitation accumulation. Generate statistics of the results.

5.1.11.1.2 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 393) section for more information.

5.1.11.1.3 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.

5.1.11.1.4 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

5.1.11.1.5 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]

PROCESS_LIST = GridStat

###
# Time Info
###

LOOP_BY = INIT
```

(continues on next page)

(continued from previous page)

```
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 12H

LEAD_SEQ = 12

LOOP_ORDER = times

###
# File I/O
###

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 =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/GridStat/GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}


###
# Field Info
###

MODEL = WRF
OBTYPE = MC_PCP

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
```

(continues on next page)

(continued from previous page)

```
###
# 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_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_ONCE_PER_FIELD = False

FCST_IS_PROB = false

FCST_GRID_STAT_PROB_THRESH = ==0.1
```

(continues on next page)

(continued from previous page)

```
OBS_IS_PROB = false

OBS_GRID_STAT_PROB_THRESH = ==0.1

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 =
#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
```

(continues on next page)

(continued from previous page)

```
#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_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 =

#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 =
```

5.1.11.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.11.1.7 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.

5.1.11.1.8 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

5.1.11.1.9 Keywords

Note:

- GridStatToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.11.2 GridStat: Using Python Embedding

met_tool_wrapper/GridStat/GridStat_python_embedding.conf

5.1.11.2.1 Scientific Objective

Compare dummy forecast data to dummy observations. Generate statistics of the results.

5.1.11.2.2 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 404) section for more information.

5.1.11.2.3 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.

5.1.11.2.4 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.

5.1.11.2.5 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

```
# GridStat METplus Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# List of applications to run - only GridStat for this case
PROCESS_LIST = GridStat
```

(continues on next page)

(continued from previous page)

```
# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG=2005080700

# End time for METplus run - must match INIT_TIME_FMT
INIT_END=2005080700

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 12

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for GridStat only
#LOG_GRID_STAT_VERBOSITY = 2

# Location of MET config file to pass to GridStat
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped
```

(continues on next page)

(continued from previous page)

```
# grid to remap data. Value is set as the 'to_grid' variable in the 'regrid' dictionary
# See MET User's Guide for more information
GRID_STAT_REGRID_TO_GRID = G130

# Name to identify model (forecast) data in output
MODEL = FCST

# Name to identify observation data in output
OBTYPE = OBS

# set the desc value in the GridStat MET config file
GRID_STAT_DESC = NA

# List of variables to compare in GridStat - FCST_VAR1 variables correspond
#  to OBS_VAR1 variables
# Note [FCST/OBS/BOTH]_GRID_STAT_VAR<n>_NAME can be used instead if different evaluations
# are needed for different tools

# Name of forecast variable 1
FCST_VAR1_NAME = {INPUT_BASE}/met_test/scripts/python/read_ascii_numpy.py {INPUT_BASE}/met_
→test/data/python/fcst.txt FCST

# Name of observation variable 1
OBS_VAR1_NAME = {INPUT_BASE}/met_test/scripts/python/read_ascii_numpy.py {INPUT_BASE}/met_
→test/data/python/obs.txt OBS

# Time relative to valid time (in seconds) to allow files to be considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
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

# MET GridStat neighborhood values
# See the MET User's Guide GridStat section for more information

# width value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_WIDTH = 1

# shape value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

# cov thresh list passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5
```

(continues on next page)

(continued from previous page)

```
# Set to true to run GridStat separately for each field specified
# Set to false to create one run of GridStat per run time that
#   includes all fields specified.
GRID_STAT_ONCE_PER_FIELD = False

# Set to true if forecast data is probabilistic
FCST_IS_PROB = false

# Only used if FCST_IS_PROB is true - sets probabilistic threshold
FCST_GRID_STAT_PROB_THRESH = ==0.1

# Set to true if observation data is probabilistic
#  Only used if configuring forecast data as the 'OBS' input
OBS_IS_PROB = false

# Only used if OBS_IS_PROB is true - sets probabilistic threshold
OBS_GRID_STAT_PROB_THRESH = ==0.1

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
GRID_STAT_NC_PAIRS_FLAG_DIFF = FALSE
GRID_STAT_NC_PAIRS_FLAG_CLIMO = FALSE
GRID_STAT_NC_PAIRS_FLAG_APPLY_MASK = FALSE


# End of [config] section and start of [dir] section
[dir]

# directory containing forecast input to GridStat
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst

# directory containing observation input to GridStat
OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/new

# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_DIR =
```

(continues on next page)

(continued from previous page)

```
# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_STDEV_INPUT_DIR =

# directory to write output from GridStat
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/GridStat_python_embedding

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to GridStat relative to FCST_GRID_STAT_INPUT_DIR
FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to look for observation input to GridStat relative to OBS_GRID_STAT_INPUT_DIR
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY


# Optional subdirectories relative to GRID_STAT_OUTPUT_DIR to write output from GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_MEAN_INPUT_
↪DIR
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_STDEV_INPUT_
↪DIR
# Not used in this example
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

# Used to specify one or more verification mask files for GridStat
# Not used for this example
GRID_STAT_VERIFICATION_MASK_TEMPLATE =
```

5.1.11.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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          = [];
```

(continues on next page)

(continued from previous page)

```
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)

```
//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.11.2.7 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

5.1.11.2.8 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.

5.1.11.2.9 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

5.1.11.2.10 Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.11.3 GridStat: Multiple Config Files Use Case

met_tool_wrapper/GridStat/GridStat_multiple_config _files.conf

5.1.11.3.1 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.

5.1.11.3.2 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 418) section for more information.

5.1.11.3.3 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.

5.1.11.3.4 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

5.1.11.3.5 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]

PROCESS_LIST = GridStat

###
# Time Info
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 12H

LEAD_SEQ = 12

LOOP_ORDER = times

###
# File I/O
###

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 =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/GridStat/GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}



###
```

(continues on next page)

(continued from previous page)

```
# Field Info
###

MODEL = WRF
OBTYPE = MC_PCP

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
###

#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_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
```

(continues on next page)

(continued from previous page)

```
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}_{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 =
#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
```

(continues on next page)

(continued from previous page)

```
#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_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 =

#GRID_STAT_MASK_GRID =
#GRID_STAT_MASK_POLY =

#GRID_STAT_DISTANCE_MAP_BADDELEY_P =
#GRID_STAT_DISTANCE_MAP_BADDELEY_MAX_DIST =
```

(continues on next page)

(continued from previous page)

```
#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

```
# GridStat Forecast Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# Name to identify model (forecast) data in output
MODEL = WRF

# List of variables to compare in GridStat - FCST_VAR1 variables correspond
#  to OBS_VAR1 variables
# Note [FCST/OBS/BOTH]_GRID_STAT_VAR<n>_NAME can be used instead if different evaluations
# are needed for different tools

# Name of forecast variable 1
FCST_VAR1_NAME = APCP

# List of levels to evaluate for forecast variable 1
# A03 = 3 hour accumulation in GRIB file
FCST_VAR1_LEVELS = A03

# List of thresholds to evaluate for each name/level combination for
#  both forecast and observation variable 1
FCST_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2

# Time relative to valid time (in seconds) to allow files to be considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0

# Set to true if forecast data is probabilistic
FCST_IS_PROB = false
```

(continues on next page)

(continued from previous page)

```
# End of [config] section and start of [dir] section
[dir]

# directory containing forecast input to GridStat
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst


# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to GridStat relative to FCST_GRID_STAT_INPUT_DIR
FCST_GRID_STAT_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH}.tm00_G212
```

GridStat_observation.conf

```
# GridStat Observation Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# Name to identify observation data in output
OBTYPE = MC_PCP

# Name of observation variable 1
OBS_VAR1_NAME = APCP_03

# List of levels to evaluate for observation variable 1
# (*,*) is NetCDF notation - must include quotes around these values!
# must be the same length as FCST_VAR1_LEVELS
OBS_VAR1_LEVELS = "(*,*)"

# List of thresholds to evaluate for each name/level combination for
#  both forecast and observation variable 1
OBS_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2

# Time relative to valid time (in seconds) to allow files to be considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
OBS_GRID_STAT_FILE_WINDOW_BEGIN = 0
OBS_GRID_STAT_FILE_WINDOW_END = 0

# Set to true if observation data is probabilistic
#  Only used if configuring forecast data as the 'OBS' input
OBS_IS_PROB = false

```

(continues on next page)

(continued from previous page)

```
# End of [config] section and start of [dir] section
[dir]

# directory containing observation input to GridStat
OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/met_test/new


# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for observation input to GridStat relative to OBS_GRID_STAT_INPUT_DIR
OBS_GRID_STAT_INPUT_TEMPLATE = ST2ml{valid?fmt=%Y%m%d%H}_A03h.nc
```

5.1.11.3.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.11.3.7 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.

5.1.11.3.8 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

5.1.11.3.9 Keywords

Note:

- GridStatToolUseCase
- MultiConfUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.12 IODA2NC

5.1.12.1 IODA2NC: Basic Use Case

met_tool_wrapper/IODA2NC/IODA2NC.conf

5.1.12.1.1 Scientific Objective

Convert IODA NetCDF files to MET NetCDF format.

5.1.12.1.2 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 425) section for more information.

5.1.12.1.3 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.

5.1.12.1.4 METplus Workflow

IODA2NC is the only tool called in this example. It processes the following run time(s):

Valid: 2020-03-10 12Z

5.1.12.1.5 METplus Configuration

parm/use_cases/met_tool_wrapper/IODA2NC/IODA2NC.conf

```
[config]

PROCESS_LIST = IODA2NC

###
# Time Info
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2020031012
VALID_END = 2020031012
VALID_INCREMENT = 6H

###
# File I/O Info
###

IODA2NC_INPUT_DIR = {INPUT_BASE}/met_test/new/ioda
```

(continues on next page)

(continued from previous page)

```
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


# OPTIONAL CONFIGURATIONS

###
# ioda2nc command line arguments
###

#IODA2NC_VALID_BEG = {valid?fmt=%Y%m%d_%H?shift=-24H}
#IODA2NC_VALID_END = {valid?fmt=%Y%m%d_%H}
#IODA2NC_NMSG = 10


###
# ioda2nc configuration variables
###

#IODA2NC_MESSAGE_TYPE =

#IODA2NC_MESSAGE_TYPE_MAP =

#IODA2NC_MESSAGE_TYPE_GROUP_MAP =

#IODA2NC_STATION_ID =

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"; },
```

(continues on next page)

(continued from previous page)

```
   { 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
```

5.1.12.1.6 MET Configuration

Note: See the *IODA2NC MET Configuration* (page 146) 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
```

(continues on next page)

(continued from previous page)

```
// 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}

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

//
// Vertical levels to retain
```

(continues on next page)

(continued from previous page)

```
//
// 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
// 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 = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_TIME_SUMMARY_DICT}

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

tmp_dir = "${MET_TMP_DIR}";

//version = "V10.0";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.12.1.7 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.

5.1.12.1.8 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

5.1.12.1.9 Keywords

Note:

- IODA2NCToolUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.13 METdbLoad

5.1.13.1 METdbLoad: Basic Use Case

met_tool_wrapper/METdbLoad/METdbLoad.conf

5.1.13.1.1 Scientific Objective

Load MET data into a database using the met_db_load.py script found in dtcenter/METdatadb

5.1.13.1.2 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 428) section for more information.

5.1.13.1.3 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.

5.1.13.1.4 METplus Workflow

METdbLoad is the only tool called in this example. It does not loop over multiple run times:

5.1.13.1.5 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]

# METdbLoad example

PROCESS_LIST = METDbLoad
```

(continues on next page)

(continued from previous page)

```
LOOP_BY = VALID

VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2005080712
VALID_END = 2005080800
VALID_INCREMENT = 12H

LOOP_ORDER = processes

MET_DB_LOAD_RUNTIME_FREQ = RUN_ONCE

MET_DATA_DB_DIR = {METPLUS_BASE}/../METdatadb

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_LOAD_STAT = true
MET_DB_LOAD_MV_LOAD_MODE = false
MET_DB_LOAD_MV_LOAD_MTD = false
MET_DB_LOAD_MV_LOAD_MPR = false

MET_DB_LOAD_INPUT_TEMPLATE = {INPUT_BASE}/met_test/out/grid_stat
```

5.1.13.1.6 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 154) 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>
```

5.1.13.1.7 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.

5.1.13.1.8 Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

5.1.13.1.9 Keywords

Note:

- METdbLoadUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.14 MODE

5.1.14.1 MODE: Using Python Embedding

met_tool_wrapper/MODE/MODE_python_embedding.conf

5.1.14.1.1 Scientific Objective

Compare dummy forecast data to dummy observations. Generate statistics of the results.

5.1.14.1.2 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.

5.1.14.1.3 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.

5.1.14.1.4 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.

5.1.14.1.5 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

```
# MODE METplus Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# List of applications to run - only MODE for this case
PROCESS_LIST = MODE

# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INIT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG=2005080700

# End time for METplus run - must match INIT_TIME_FMT
INIT_END=2005080700

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 12

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
```

(continues on next page)

(continued from previous page)

```
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for MODE only
#LOG_MODE_VERBOSITY = 2

# Location of MET config file to pass to the MODE
# References CONFIG_DIR from the [dir] section
MODE_CONFIG_FILE = {CONFIG_DIR}/MODEConfig_wrapped

# grid to remap data. Value is set as the 'to_grid' variable in the 'regrid' dictionary
# See MET User's Guide for more information
MODE_REGRID_TO_GRID = NONE

MODE_MASK_MISSING_FLAG = BOTH

MODE_OUTPUT_PREFIX = FCST_vs_OBS

# Location of merge config file to pass to the MODE
# References CONFIG_DIR from the [dir] section
# Not used if unset or set to an empty string
MODE_MERGE_CONFIG_FILE =

# Name to identify model (forecast) data in output
MODEL = WRF

# Name to identify observation data in output
OBTYPE = WRF

#MODE_GRID_RES = 4

# turn on quilting
MODE_QUILT = True

# convolution radius list
FCST_MODE_CONV_RADIUS = 5

# convolution radius list
OBS_MODE_CONV_RADIUS = 5

# convolution threshold list
FCST_MODE_CONV_THRESH = >=80.0
```

(continues on next page)

(continued from previous page)

```
# convolution threshold list
OBS_MODE_CONV_THRESH = >=80.0

# merge threshold list
FCST_MODE_MERGE_THRESH = >=75.0

# merge threshold list
OBS_MODE_MERGE_THRESH = >=75.0

# merge flag: options are NONE, THRESH, ENGINE, or BOTH
FCST_MODE_MERGE_FLAG = NONE

# merge flag: options are NONE, THRESH, ENGINE, or BOTH
OBS_MODE_MERGE_FLAG = NONE

# List of variables to compare in MODE - FCST_VAR1 variables correspond
# to OBS_VAR1 variables

# Name of forecast variable 1
FCST_VAR1_NAME = {INPUT_BASE}/met_test/scripts/python/read_ascii_numpy.py {INPUT_BASE}/met_
↪test/data/python/fcst.txt FCST

# Name of observation variable 1
OBS_VAR1_NAME = {INPUT_BASE}/met_test/scripts/python/read_ascii_numpy.py {INPUT_BASE}/met_
↪test/data/python/obs.txt OBS

# List of levels to evaluate for observation variable 1
# P500 = 500 mb pressure level in GRIB file
# must be the same length as FCST_VAR1_LEVELS
OBS_VAR1_LEVELS = P500

# Time relative to valid time (in seconds) to allow files to be considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
FCST_MODE_FILE_WINDOW_BEGIN = 0
FCST_MODE_FILE_WINDOW_END = 0
OBS_MODE_FILE_WINDOW_BEGIN = 0
OBS_MODE_FILE_WINDOW_END = 0

# Set to true if forecast data is probabilistic
FCST_IS_PROB = false

# Set to true if observation data is probabilistic
#  Only used if configuring forecast data as the 'OBS' input
```

(continues on next page)

(continued from previous page)

```
OBS_IS_PROB = false

# End of [config] section and start of [dir] section
[dir]

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/met_config

# directory containing forecast input to MODE
FCST_MODE_INPUT_DIR =

# directory containing observation input to MODE
OBS_MODE_INPUT_DIR =

# directory to write output from MODE
MODE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/MODE_python_embedding

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to MODE relative to FCST_MODE_INPUT_DIR
FCST_MODE_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to look for observation input to MODE relative to OBS_MODE_INPUT_DIR
OBS_MODE_INPUT_TEMPLATE = PYTHON_NUMPY

# Optional subdirectories relative to MODE_OUTPUT_DIR to write output from MODE
MODE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}

# Used to specify a verification mask file for MODE
# Not used for this example.
MODE_VERIFICATION_MASK_TEMPLATE =
```

5.1.14.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *MODE MET Configuration* (page 159) 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}

//
// 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}
}

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}
}
```

(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}
```

5.1.14.1.7 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

5.1.14.1.8 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.

5.1.14.1.9 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

5.1.14.1.10 Keywords

Note:

- MODEToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.14.2 MODE: Basic Use Case

met_tool_wrapper/MODE/MODE.conf

5.1.14.2.1 Scientific Objective

Compare relative humidity 12 hour forecast to 0 hour observations. Generate statistics of the results.

5.1.14.2.2 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 452) section for more information.

5.1.14.2.3 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.

5.1.14.2.4 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

5.1.14.2.5 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]

PROCESS_LIST = MODE

LOOP_ORDER = times
LOOP_BY = INIT

INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2005080700
INIT_END=2005080700
INIT_INCREMENT = 12H

LEAD_SEQ = 12

#LOG_MODE_VERBOSITY = 2

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

MODE_OUTPUT_DIR = {OUTPUT_BASE}/mode
MODE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}


MODEL = WRF

MODE_DESC = NA

OBTYPE = WRF

MODE_CONFIG_FILE = {PARM_BASE}/met_config/MODEConfig_wrapped


#MODE_FCST_FILE_TYPE =

FCST_VAR1_NAME = RH
FCST_VAR1_LEVELS = P500

#MODE_MULTIVAR_LOGIC =
```

(continues on next page)

(continued from previous page)

```
FCST_MODE_CONV_RADIUS = 5
FCST_MODE_CONV_THRESH = >=80.0
FCST_MODE_MERGE_THRESH = >=75.0
FCST_MODE_MERGE_FLAG = NONE

#MODE_FCST_FILTER_ATTR_NAME =
#MODE_FCST_FILTER_ATTR_THRESH =
#MODE_FCST_CENSOR_THRESH =
#MODE_FCST_CENSOR_VAL =
#MODE_FCST_VLD_THRESH =

FCST_IS_PROB = false


#MODE_OBS_FILE_TYPE =

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_FILTER_ATTR_NAME =
#MODE_OBS_FILTER_ATTR_THRESH =
#MODE_OBS_CENSOR_THRESH =
#MODE_OBS_CENSOR_VAL =
#MODE_OBS_VLD_THRESH =

OBS_IS_PROB = false


FCST_MODE_FILE_WINDOW_BEGIN = 0
FCST_MODE_FILE_WINDOW_END = 0
OBS_MODE_FILE_WINDOW_BEGIN = 0
OBS_MODE_FILE_WINDOW_END = 0

MODE_REGRID_TO_GRID = NONE
#MODE_REGRID_METHOD =
#MODE_REGRID_WIDTH =
#MODE_REGRID_VLD_THRESH =
#MODE_REGRID_SHAPE =
```

(continues on next page)

(continued from previous page)

```
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 =
#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 =
```

5.1.14.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *MODE MET Configuration* (page 159) 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 = {
```

(continues on next page)

(continued from previous page)

```
${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}

//
// 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}
}

obs = {
   ${METPLUS_OBS_FIELD}
```

(continues on next page)

(continued from previous page)

```
   ${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}
}

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

//
// 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
//
```

(continues on next page)

(continued from previous page)

```
//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 = (
      (    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;
```

(continues on next page)

(continued from previous page)

```
   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;
}

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
//
```

(continues on next page)

(continued from previous page)

```
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";

tmp_dir = "${MET_TMP_DIR}";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.14.2.7 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.

5.1.14.2.8 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
```

5.1.14.2.9 Keywords

Note:

- MODEToolUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.15 MTD

5.1.15.1 MTD using Python Embedding

met_tool_wrapper/MTD/MTD_python_embedding.conf

5.1.15.1.1 Scientific Objective

Compare forecast and observation 3 hour precipitation accumulation spatially and temporally over the 6 hour, 9 hour, and 12 hour forecast leads.

5.1.15.1.2 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 463) section for more information.

5.1.15.1.3 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.

5.1.15.1.4 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.

5.1.15.1.5 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

```
# MTD (MODE Time Domain) using Python Embedding METplus Configuration

[config]

# List of applications to run - only MTD for this case
PROCESS_LIST = MTD

# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG=2005080700

# End time for METplus run - must match INIT_TIME_FMT
INIT_END=2005080700

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
```

(continues on next page)

(continued from previous page)

```
INIT_INCREMENT=1M

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0, 1, 2


# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# if true, only process a single data set with MTD
MTD_SINGLE_RUN = False

# Data to process in single mode
# FCST and OBS are valid options
MTD_SINGLE_DATA_SRC = OBS

# forecast convolution radius list
FCST_MTD_CONV_RADIUS = 0

# forecast convolution threshold list
FCST_MTD_CONV_THRESH = >=10

# observation convolution radius list
OBS_MTD_CONV_RADIUS = 15

# observation convolution threshold list
OBS_MTD_CONV_THRESH = >=1.0

# list of variables to compare
FCST_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/read_ascii_numpy.py MET_PYTHON_INPUT_ARG␣
↪FCST

OBS_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/read_ascii_numpy.py MET_PYTHON_INPUT_ARG␣
↪OBS

FCST_MTD_INPUT_DATATYPE = PYTHON_NUMPY
```

(continues on next page)

(continued from previous page)

```
OBS_MTD_INPUT_DATATYPE = PYTHON_NUMPY


# description of data to be processed
# used in output file path
MODEL = FCST
OBTYPE = OBS

#MTD_DESC =

# location of MODE Time Domain MET config file
# References CONFIG_DIR from the [dir] section
MTD_CONFIG_FILE = {CONFIG_DIR}/MTDConfig_wrapped

# grid to remap data. Value is set as the 'to_grid' variable in the 'regrid' dictionary
# See MET User's Guide for more information
MTD_REGRID_TO_GRID = OBS

# Minimum volume
FCST_MTD_MIN_VOLUME = 2000

# convolution radius for forecast data
FCST_MTD_CONV_RADIUS = 15

# convolution threshold for forecast data
FCST_MTD_CONV_THRESH = >=5.0

# output prefix to add to output filenames
MTD_OUTPUT_PREFIX = PYTHON

# set to True if forecast data is probabilistic
FCST_IS_PROB = false

# True if probabilistic information is in the GRIB Product Definition Section
FCST_PROB_IN_GRIB_PDS = false

# End of [config] section and start of [dir] section
[dir]
# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/met_config

# input and output data directories for each application in PROCESS_LIST
FCST_MTD_INPUT_DIR = {INPUT_BASE}/met_test/data/python
```

(continues on next page)

(continued from previous page)

```
OBS_MTD_INPUT_DIR = {INPUT_BASE}/met_test/data/python

MTD_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/MTD/mtd_python_embedding

# End of [dir] section and start of [filename_templates] section
[filename_templates]
# format of filenames

FCST_MTD_INPUT_TEMPLATE= fcst.txt

OBS_MTD_INPUT_TEMPLATE = obs.txt

MTD_OUTPUT_TEMPLATE =
```

5.1.15.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *MTD MET Configuration* (page 172) 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 )
```

(continues on next page)

(continued from previous page)

```
      (  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 )

   );

   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 )
```

(continues on next page)

(continued from previous page)

```
      (  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;
   cluster_id   = true;

}

txt_output = {

   attributes_2d   = true;
   attributes_3d   = true;

}


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

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

(continues on next page)

(continued from previous page)

```
tmp_dir = "${MET_TMP_DIR}";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.15.1.7 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

5.1.15.1.8 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.

5.1.15.1.9 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

5.1.15.1.10 Keywords

Note:

- MTDToolUseCase
- PythonEmbeddingFileUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.15.2 Basic MTD Use Case

met_tool_wrappper/MTD/MTD.conf

5.1.15.2.1 Scientific Objective

Compare forecast and observation 3 hour precipitation accumulation spatially and temporally over the 6 hour, 9 hour, and 12 hour forecast leads.

5.1.15.2.2 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 474) section for more information.

5.1.15.2.3 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.

5.1.15.2.4 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

5.1.15.2.5 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

```
# MTD (MODE Time Domain) METplus Configuration

[config]
```

(continues on next page)

(continued from previous page)

```
# List of applications to run - only MTD for this case
PROCESS_LIST = MTD

# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG=2005080700

# End time for METplus run - must match INIT_TIME_FMT
INIT_END=2005080700

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT=1M

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 6H, 9H, 12H


# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# if true, only process a single data set with MTD
MTD_SINGLE_RUN = False
```

(continues on next page)

(continued from previous page)

```
# Data to process in single mode
# FCST and OBS are valid options
MTD_SINGLE_DATA_SRC = OBS

# forecast convolution radius list
FCST_MTD_CONV_RADIUS = 10

# forecast convolution threshold list
FCST_MTD_CONV_THRESH = >=0.0

# observation convolution radius list
OBS_MTD_CONV_RADIUS = 10

# observation convolution threshold list
OBS_MTD_CONV_THRESH = >=0.0

# list of variables to compare
FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = A03
FCST_VAR1_THRESH = gt12.7

OBS_VAR1_NAME = APCP_03
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_THRESH = gt12.7

# description of data to be processed
# used in output file path
MODEL = WRF
MTD_DESC = NA
OBTYPE = MC_PCP

# location of MODE Time Domain MET config file
# References CONFIG_DIR from the [dir] section
MTD_CONFIG_FILE = {CONFIG_DIR}/MTDConfig_wrapped

# grid to remap data. Value is set as the 'to_grid' variable in the 'regrid' dictionary
# See MET User's Guide for more information
MTD_REGRID_TO_GRID = OBS

# Minimum volume
MTD_MIN_VOLUME = 2000

# output prefix to add to output filenames
MTD_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}_{CURRENT_FCST_
↪LEVEL}
```

(continues on next page)

(continued from previous page)

```
# set to True if forecast data is probabilistic
FCST_IS_PROB = False

# True if probabilistic information is in the GRIB Product Definition Section
FCST_PROB_IN_GRIB_PDS = false


# End of [config] section and start of [dir] section
[dir]
# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/met_config

# input and output data directories for each application in PROCESS_LIST
FCST_MTD_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst

OBS_MTD_INPUT_DIR = {INPUT_BASE}/met_test/new

MTD_OUTPUT_DIR = {OUTPUT_BASE}/mtd

# End of [dir] section and start of [filename_templates] section
[filename_templates]
# format of filenames

FCST_MTD_INPUT_TEMPLATE= {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%HH}.tm00_G212

OBS_MTD_INPUT_TEMPLATE = ST2ml{valid?fmt=%Y%m%d%H}_A03h.nc

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

5.1.15.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *MTD MET Configuration* (page 172) 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;

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

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

(continues on next page)

(continued from previous page)

```
   //

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 = (

      ( -3.0, 0.0 )
```

(continues on next page)

(continued from previous page)

```
    ( -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;
   cluster_id   = true;
```

(continues on next page)

(continued from previous page)

```
}

txt_output = {

   attributes_2d   = true;
   attributes_3d   = true;

}


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

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

tmp_dir = "${MET_TMP_DIR}";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.15.2.7 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.

5.1.15.2.8 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
```

5.1.15.2.9 Keywords

Note:

- MTDToolUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.16 PB2NC

5.1.16.1 PB2NC: Basic Use Case

met_tool_wrapper/PB2NC/PB2NC.conf

5.1.16.1.1 Scientific Objective

Simply converting file formats so point observations can be read by the MET tools.

5.1.16.1.2 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 481) section for more information.

Data Source: Unknown

5.1.16.1.3 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.

5.1.16.1.4 METplus Workflow

PB2NC is the only tool called in this example. It processes the following run time:

Valid: 2007-03-31_12Z

5.1.16.1.5 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]

PROCESS_LIST = PB2NC

```

(continues on next page)

(continued from previous page)

```
LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2007033112
VALID_END = 2007033112
VALID_INCREMENT = 1M

LEAD_SEQ = 0

PB2NC_OFFSETS = 12

PB2NC_SKIP_IF_OUTPUT_EXISTS = True

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_OUTPUT_DIR = {OUTPUT_BASE}/pb2nc
PB2NC_OUTPUT_TEMPLATE = sample_pb.nc


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 =
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
```

(continues on next page)

(continued from previous page)

```
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 =
```

5.1.16.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *PB2NC MET Configuration* (page 181) 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}

//
```

(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}
```

5.1.16.1.7 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.

5.1.16.1.8 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

5.1.16.1.9 Keywords

Note:

- PB2NCToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.17 PCPCombine

5.1.17.1 PCPCombine: Custom String Looping Use Case

met_tool_wrapper/PCPCombine/PCPCombine_loop_custom.conf

5.1.17.1.1 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.

5.1.17.1.2 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 485) section for more information.

Data Source: WRF-AFW

5.1.17.1.3 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.

5.1.17.1.4 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

5.1.17.1.5 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]

PROCESS_LIST = PCPCombine

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2009123112
INIT_END = 2009123112
INIT_INCREMENT = 1M

LEAD_SEQ = 24H

LOOP_ORDER = times

PCP_COMBINE_CUSTOM_LOOP_LIST = arw-fer-gep1, arw-fer-gep5, arw-sch-gep2, arw-sch-gep6, arw-
↪tom-gep3, arw-tom-gep7

#LOG_PCP_COMBINE_VERBOSITY = 2

FCST_PCP_COMBINE_RUN = True
FCST_PCP_COMBINE_METHOD = ADD

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

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
```

(continues on next page)

(continued from previous page)

```
FCST_PCP_COMBINE_OUTPUT_NAME = APCP
```

5.1.17.1.6 MET Configuration

None. PCPCombine does not use configuration files.

5.1.17.1.7 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.

5.1.17.1.8 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

5.1.17.1.9 Keywords

Note:

- PCPCombineToolUseCase
- CustomStringLoopingUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.17.2 PCPCombine: Bucket Interval Use Case

met_tool_wrapper/PCPCombine/PCPCombine_bucket.conf

5.1.17.2.1 Scientific Objective

Build a 15 hour precipitation accumulation field from varying accumulation fields.

5.1.17.2.2 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 489) section for more information.

Data Source: GFS

5.1.17.2.3 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.

5.1.17.2.4 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

5.1.17.2.5 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]

PROCESS_LIST = PcpCombine

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2012040900
INIT_END = 2012040900
INIT_INCREMENT = 86400

LEAD_SEQ = 15H

LOOP_ORDER = times

FCST_PCP_COMBINE_RUN = True
FCST_PCP_COMBINE_METHOD = ADD

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

#LOG_PCP_COMBINE_VERBOSITY = 2

FCST_PCP_COMBINE_MAX_FORECAST = 2d

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
```

5.1.17.2.6 MET Configuration

None. PCPCombine does not use configuration files.

5.1.17.2.7 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.

5.1.17.2.8 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

5.1.17.2.9 Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.17.3 PCPCombine: User-defined Command Use Case

met_tool_wrapper/PCPCombine/PCPCombine_user_defined.conf

5.1.17.3.1 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.

5.1.17.3.2 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 492) section for more information.

Data Source: WRF

5.1.17.3.3 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.

5.1.17.3.4 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

5.1.17.3.5 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]

PROCESS_LIST = PCPCombine

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2005080700
INIT_END = 2005080700
INIT_INCREMENT = 1M

LEAD_SEQ = 24H

LOOP_ORDER = times

FCST_PCP_COMBINE_RUN = True
FCST_PCP_COMBINE_METHOD = USER_DEFINED

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
```

(continues on next page)

(continued from previous page)

```
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


FCST_PCP_COMBINE_COMMAND = -derive sum,min,max,range,mean,stdev,vld_count {FCST_PCP_COMBINE_
↪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
```

5.1.17.3.6 MET Configuration

None. PCPCombine does not use configuration files.

5.1.17.3.7 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.

5.1.17.3.8 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

5.1.17.3.9 Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.17.4 PCPCombine: ADD Use Case

met_tool_wrapper/PCPCombine/PCPCombine_add.conf

5.1.17.4.1 Scientific Objective

Build a 15 minute precipitation accumulation field from 5 minute accumulation fields.

5.1.17.4.2 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 496) section for more information.

Data Source: NEWSe

5.1.17.4.3 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.

5.1.17.4.4 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

5.1.17.4.5 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]

PROCESS_LIST = PCPCombine

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H%M
VALID_BEG = 201908021815
VALID_END = 201908021815
VALID_INCREMENT = 1M

LEAD_SEQ = 15M

LOOP_ORDER = times

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


#LOG_PCP_COMBINE_VERBOSITY = 2

FCST_PCP_COMBINE_RUN = True
FCST_PCP_COMBINE_METHOD = ADD

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
```

5.1.17.4.6 MET Configuration

None. PCPCombine does not use configuration files.

5.1.17.4.7 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.

5.1.17.4.8 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

5.1.17.4.9 Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.17.5 PCPCombine: Python Embedding Use Case

met_tool_wrapper/PCPCombine/PCPCombine_python_embedding.conf

5.1.17.5.1 Scientific Objective

Build a 2 hour precipitation accumulation field from 30 minute IMERG data.

5.1.17.5.2 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 500) section for more information.

Data Source: IMERG

5.1.17.5.3 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

5.1.17.5.4 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.

5.1.17.5.5 METplus Workflow

PCPCombine is the only tool called in this example. It processes the following run times:

Valid: 2018-01-02_13:30Z

5.1.17.5.6 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]

PROCESS_LIST = PCPCombine

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H%M
VALID_BEG=201801021300
```

(continues on next page)

(continued from previous page)

```
VALID_END=201801021300
VALID_INCREMENT=43200

LEAD_SEQ = 0

LOOP_ORDER = times

#LOG_PCP_COMBINE_VERBOSITY = 2

OBS_PCP_COMBINE_RUN = True
OBS_PCP_COMBINE_METHOD = ADD

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


OBS_VAR1_NAME = APCP
OBS_VAR1_LEVELS = A06

OBS_PCP_COMBINE_INPUT_DATATYPE = PYTHON_NUMPY
OBS_PCP_COMBINE_INPUT_ACCUMS = 6
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
```

5.1.17.5.7 MET Configuration

None. PCPCombine does not use configuration files.

5.1.17.5.8 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.

5.1.17.5.9 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:

- .

5.1.17.5.10 Keywords

Note:

- PCPCombineToolUseCase
- PythonEmbeddingFileUseCase
- MET_PYTHON_EXEUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.17.6 PCPCombine: SUM Use Case

met_tool_wrapper/PCPCombine/PCPCombine_sum.conf

5.1.17.6.1 Scientific Objective

Build a 15 minute precipitation accumulation field from 5 minute accumulation fields.

5.1.17.6.2 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 503) section for more information.

Data Source: NEWSe

5.1.17.6.3 METplus Components

This use case utilizes the METplus PCPCombine wrapper to build a command that will look for valid data to build an accumulation.

5.1.17.6.4 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

5.1.17.6.5 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]

PROCESS_LIST = PCPCombine

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H%M
VALID_BEG = 201908021815
VALID_END = 201908021815
VALID_INCREMENT = 1M

LEAD_SEQ = 15M

LOOP_ORDER = times

FCST_PCP_COMBINE_RUN = True
FCST_PCP_COMBINE_METHOD = SUM

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

#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
```

5.1.17.6.6 MET Configuration

None. PCPCombine does not use configuration files.

5.1.17.6.7 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.

5.1.17.6.8 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

5.1.17.6.9 Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.17.7 PCPCombine: SUBTRACT Use Case

met_tools_wrapper/PCPCombine/PCPCombine_subtract.conf

5.1.17.7.1 Scientific Objective

Extract a 3 hour precipitation accumulation field by subtracting a 15 hour accumulation field from an 18 hour accumulation field.

5.1.17.7.2 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 506) section for more information.

Data Source: WRF

5.1.17.7.3 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.

5.1.17.7.4 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

5.1.17.7.5 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]

PROCESS_LIST = PCPCombine

LOOP_BY = INIT

INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2005080700
INIT_END = 2005080700
INIT_INCREMENT = 1M

LEAD_SEQ = 18H

LOOP_ORDER = times
```

(continues on next page)

(continued from previous page)

```
#LOG_PCP_COMBINE_VERBOSITY = 2

FCST_PCP_COMBINE_RUN = True
FCST_PCP_COMBINE_METHOD = SUBTRACT

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

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
```

5.1.17.7.6 MET Configuration

None. PCPCombine does not use configuration files.

5.1.17.7.7 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.

5.1.17.7.8 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

5.1.17.7.9 Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.17.8 PCPCombine: DERIVE Use Case

met_tool_wrapper/PCPCombine/PCPCombine_derive.conf

5.1.17.8.1 Scientific Objective

Derive statistics (sum, minimum, maximum, range, mean, standard deviation, and valid count) using six 3 hour precipitation accumulation fields.

5.1.17.8.2 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 510) section for more information.

Data Source: WRF

5.1.17.8.3 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.

5.1.17.8.4 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

5.1.17.8.5 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]

PROCESS_LIST = PCPCombine

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2005080700
INIT_END = 2005080700
INIT_INCREMENT = 1M

LEAD_SEQ = 24H

LOOP_ORDER = times

#LOG_PCP_COMBINE_VERBOSITY = 2

FCST_PCP_COMBINE_RUN = True
FCST_PCP_COMBINE_METHOD = DERIVE

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

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 =
```

(continues on next page)

(continued from previous page)

```
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 =
```

5.1.17.8.6 MET Configuration

None. PCPCombine does not use configuration files.

5.1.17.8.7 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.

5.1.17.8.8 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

5.1.17.8.9 Keywords

Note:

- PCPCombineToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.18 PlotDataPlane

5.1.18.1 PlotDataPlane: Python Embedding Input

met_tool_wrapper/PlotDataPlane/PlotDataPlane_python_embedding.conf

5.1.18.1.1 Scientific Objective

Generate a postscript image to test if the input data can be read by the MET tools.

5.1.18.1.2 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

5.1.18.1.3 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.

5.1.18.1.4 METplus Workflow

PlotDataPlane is the only tool called in this example. It processes the following run time:

Valid: 2005-08-07 12Z

5.1.18.1.5 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]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - only PlotDataPlane for this case
PROCESS_LIST = PlotDataPlane

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 2005080712
```

(continues on next page)

(continued from previous page)

```
# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 2005080712

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 0

LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for PlotDataPlane only
LOG_PLOT_DATA_PLANE_VERBOSITY = 1

PLOT_DATA_PLANE_FIELD_NAME = {MET_INSTALL_DIR}/share/met/python/read_ascii_numpy.py {INPUT_
↪BASE}/met_test/data/python/fcst.txt FCST

PLOT_DATA_PLANE_TITLE = Python Embedding FCST

PLOT_DATA_PLANE_COLOR_TABLE =

PLOT_DATA_PLANE_RANGE_MIN_MAX =


# End of [config] section and start of [dir] section
[dir]

# Input/Output directories can be left empty if the corresponding template contains the full␣
↪path to the files
PLOT_DATA_PLANE_INPUT_DIR =
PLOT_DATA_PLANE_OUTPUT_DIR =

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for input to PlotDataPlane relative to PLOT_DATA_PLANE_INPUT_DIR
PLOT_DATA_PLANE_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to use to write output from PlotDataPlane
PLOT_DATA_PLANE_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/plot_data_plane/py_embed_
↪fcst.ps
```

5.1.18.1.6 MET Configuration

This tool does not use a MET configuration file.

5.1.18.1.7 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

5.1.18.1.8 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.

5.1.18.1.9 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

5.1.18.1.10 Keywords

Note:

- PlotDataPlaneToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.18.2 PlotDataPlane: NetCDF Input

met_tool_wrapper/PlotDataPlane/PlotDataPlane_netcdf.conf

5.1.18.2.1 Scientific Objective

Generate a postscript image to test if the input data can be read by the MET tools.

5.1.18.2.2 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

5.1.18.2.3 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.

5.1.18.2.4 METplus Workflow

PlotDataPlane is the only tool called in this example. It processes the following run time:

Valid: 2007-03-30 0Z

5.1.18.2.5 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]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - only PlotDataPlane for this case
PROCESS_LIST = PlotDataPlane

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 2005080712
```

(continues on next page)

(continued from previous page)

```
# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 2005080712

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 0

LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for PlotDataPlane only
LOG_PLOT_DATA_PLANE_VERBOSITY = 1

PLOT_DATA_PLANE_FIELD_NAME = APCP_12
PLOT_DATA_PLANE_FIELD_LEVEL = "(*,*)"

PLOT_DATA_PLANE_TITLE = NC MET 12-hour APCP

PLOT_DATA_PLANE_COLOR_TABLE =

PLOT_DATA_PLANE_RANGE_MIN_MAX = 1.0 3.0


# End of [config] section and start of [dir] section
[dir]

# Input/Output directories can be left empty if the corresponding template contains the full
↪path to the files
PLOT_DATA_PLANE_INPUT_DIR =
PLOT_DATA_PLANE_OUTPUT_DIR =

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for input to PlotDataPlane relative to 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
# Template to use to write output from PlotDataPlane
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
```

5.1.18.2.6 MET Configuration

This tool does not use a MET configuration file.

5.1.18.2.7 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.

5.1.18.2.8 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

5.1.18.2.9 Keywords

Note:

- PlotDataPlaneToolUseCase
- NetCDFFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.18.3 PlotDataPlane: GRIB1 Input

met_tool_wrapper/PlotDataPlane/PlotDataPlane_grib1.conf

5.1.18.3.1 Scientific Objective

Generate a postscript image to test if the input data can be read by the MET tools.

5.1.18.3.2 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

5.1.18.3.3 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.

5.1.18.3.4 METplus Workflow

PlotDataPlane is the only tool called in this example. It processes the following run time:

Valid: 2007-03-30 0Z

5.1.18.3.5 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]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - only PlotDataPlane for this case
PROCESS_LIST = PlotDataPlane

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 20070330

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 20070330
```

(continues on next page)

(continued from previous page)

```
# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 0

LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for PlotDataPlane only
LOG_PLOT_DATA_PLANE_VERBOSITY = 1

PLOT_DATA_PLANE_FIELD_NAME = TMP
PLOT_DATA_PLANE_FIELD_LEVEL = Z2

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 =

# End of [config] section and start of [dir] section
[dir]

# Input/Output directories can be left empty if the corresponding template contains the full␣
↪path to the files
PLOT_DATA_PLANE_INPUT_DIR =
PLOT_DATA_PLANE_OUTPUT_DIR =

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for input to PlotDataPlane relative to 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

# Template to use to write output from PlotDataPlane
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
```

5.1.18.3.6 MET Configuration

This tool does not use a MET configuration file.

5.1.18.3.7 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.

5.1.18.3.8 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

5.1.18.3.9 Keywords

Note:

- PlotDataPlaneToolUseCase
- GRIBFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.19 Point2Grid

5.1.19.1 Point2Grid: Basic Use Case

met_tool_wrapper/Point2Grid/Point2Grid.conf

5.1.19.1.1 Scientific Objective

Putting point observations onto a grid for use with other tools.

5.1.19.1.2 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 527) section for more information.

5.1.19.1.3 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.

5.1.19.1.4 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

5.1.19.1.5 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]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - only Point2Grid for this case
PROCESS_LIST = Point2Grid

# time looping - 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
LOOP_BY = INIT

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG = 2017060100

# End time for METplus run - must match INIT_TIME_FMT
```

(continues on next page)

(continued from previous page)

```
INIT_END = 2017060300

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 24H

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 12H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for Point2Grid only
# This has shown to be a bad idea because it takes precedence over LOG_MET_VERBOSITY/LOG_
↪VERBOSIT
# LOG_POINT2GRID_VERBOSITY = 1

# Time relative to valid time (in seconds if no units are specified) to allow files to be␣
↪considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
POINT2GRID_FILE_WINDOW_BEGIN = 0
POINT2GRID_FILE_WINDOW_END = 0

# Value to pass with the -to_grid See MET User's Guide for more information
POINT2GRID_REGRID_TO_GRID = G212

# Value to pass with the -field string. See MET User's Guide for more information
# FIELD and LEVEL both end up in the -field string
POINT2GRID_INPUT_FIELD =TMP
POINT2GRID_INPUT_LEVEL =

# Value to pass with the -qc argument
POINT2GRID_QC_FLAGS = 0

# Value to pass with the -adp argument - This is a file name with GOES Aerosol Detection␣
↪Product data
```

(continues on next page)

(continued from previous page)

```
POINT2GRID_ADP =

# Value to pass with the -method argumen - Default is UW_MEAN, other examples are
POINT2GRID_REGRID_METHOD = MAX

# Value to pass with the -gaussian-dx argument - Distance interval for gaussian smoothing
# Default is 81.271
POINT2GRID_GAUSSIAN_DX = 81.271

# Value to pass with the -gaussian-radius argument - radius of influence for the gaussian␣
↪smoothing
# Default is 120
POINT2GRID_GAUSSIAN_RADIUS = 120

# Value to pass with the -prob_cat_thresh argument - threshold for probability of occurrence
POINT2GRID_PROB_CAT_THRESH =

# Value to pass with the -vld_thresh argument - threshold for percentage of valid data .5␣
↪default
POINT2GRID_VLD_THRESH =

# End of [config] section and start of [dir] section
[dir]

# Input/Output directories can be left empty if the corresponding template contains the full␣
↪path to the files
POINT2GRID_INPUT_DIR =

POINT2GRID_OUTPUT_DIR =

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for input to Point2Grid relative to POINT2GRID_INPUT_DIR
POINT2GRID_INPUT_TEMPLATE = {INPUT_BASE}/met_test/data/sample_obs/prepbufr/sample_pb.nc

# Template to use to write output from Point2Grid
POINT2GRID_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/Point2Grid/grid.{init?fmt=%Y%d%H}
↪.nc
```

5.1.19.1.6 MET Configuration

None. Point2Grid does not use configuration files.

5.1.19.1.7 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.

5.1.19.1.8 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

5.1.19.1.9 Keywords

Note:

- Point2GridToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.20 PointStat

5.1.20.1 PointStat: Basic Use Case

met_tool_wrapper/PointStat/PointStat.conf

5.1.20.1.1 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.

5.1.20.1.2 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 537) section for more information.

Data Source: Unknown

5.1.20.1.3 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.

5.1.20.1.4 METplus Workflow

PointStat is the only tool called in this example. It processes the following run times:

Valid: 2007-03-30_0Z

5.1.20.1.5 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]

PROCESS_LIST = PointStat

###
# Time Info
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20070330
INIT_END = 20070330
INIT_INCREMENT = 1M

LEAD_SEQ = 36

###
# File I/O
###

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
```

(continues on next page)

(continued from previous page)

```
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
###

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
###

#LOG_POINT_STAT_VERBOSITY = 2
```

(continues on next page)

(continued from previous page)

```
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 =
#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
```

(continues on next page)

(continued from previous page)

```
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 =

#POINT_STAT_MESSAGE_TYPE_GROUP_MAP =
```

5.1.20.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *PointStat MET Configuration* (page 194) 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_GRID}
   ${METPLUS_MASK_POLY}
   ${METPLUS_MASK_SID}
   //llpnt =
   ${METPLUS_MASK_LLPNT}
}

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

//
// 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     = "";
}

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

//
// Interpolation methods
//
//interp = {
${METPLUS_INTERP_DICT}

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

//
// HiRA verification method
//
//hira = {
${METPLUS_HIRA_DICT}

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

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

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

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.20.1.7 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.

5.1.20.1.8 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

5.1.20.1.9 Keywords

Note:

- PointStatToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.20.2 PointStat: Using Python Embedding

met_tool_wrapper/PointStat/PointStat_python_embedding.conf

5.1.20.2.1 Scientific Objective

Read forecast data using a Python embedding script. Compare with point observation data.

5.1.20.2.2 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

5.1.20.2.3 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

5.1.20.2.4 METplus Workflow

PointStat is the only tool called in this example. It processes the following run times:

Valid: 2020-09-06 6Z

5.1.20.2.5 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]

# List of applications to run - only PointStat for this case
PROCESS_LIST = PointStat

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 2020090606

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 2020090606
```

(continues on next page)

(continued from previous page)

```
# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# Verbosity of MET output - overrides LOG_VERBOSITY for PointStat only
#LOG_POINT_STAT_VERBOSITY = 2

# Location of MET config file to pass to GridStat
# References PARM_BASE which is the location of the parm directory corresponding
# to the ush directory of the master_metplus.py script that is called
# or the value of the environment variable METPLUS_PARM_BASE if set
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

# Time relative to each input file's valid time (in seconds if no units are specified) for
↪data within the file to be
# considered valid. Values are set in the 'obs_window' dictionary in the PointStat config
↪file
OBS_POINT_STAT_WINDOW_BEGIN = -5400
OBS_POINT_STAT_WINDOW_END = 5400

# Optional list of offsets to look for point observation data
POINT_STAT_OFFSETS = 0
```

(continues on next page)

(continued from previous page)

```
# Model/fcst and obs name, e.g. GFS, NAM, GDAS, etc.
#MODEL =

#POINT_STAT_DESC = NA

#POINT_STAT_REGRID_TO_GRID =
#POINT_STAT_REGRID_METHOD =
#POINT_STAT_REGRID_WIDTH =

POINT_STAT_OUTPUT_PREFIX =

# sets the -obs_valid_beg command line argument (optional)
# not used for this example
#POINT_STAT_OBS_VALID_BEG = {valid?fmt=%Y%m%d_%H}

# sets the -obs_valid_end command line argument (optional)
# not used for this example
#POINT_STAT_OBS_VALID_END = {valid?fmt=%Y%m%d_%H}

# Verification Masking regions
# Indicate which grid and polygon masking region, if applicable
POINT_STAT_GRID = FULL

# List of full path to poly masking files.  NOTE: Only short lists of poly
# files work (those that fit on one line), a long list will result in an
# environment variable that is too long, resulting in an error.  For long
# lists of poly masking files (i.e. all the mask files in the NCEP_mask
# directory), define these in the MET point_stat configuration file.
POINT_STAT_POLY =
POINT_STAT_STATION_ID =

# Message types, if all message types are to be returned, leave this empty,
# otherwise indicate the message types of interest.
POINT_STAT_MESSAGE_TYPE = ADPUPA

# Variables and levels as specified in the field dictionary of the MET
# point_stat configuration file. Specify as FCST_VARn_NAME, FCST_VARn_LEVELS,
# (optional) FCST_VARn_OPTION

# set to True to run PointStat once for each name/level combination
# set to False to run PointStat once per run time including all fields
POINT_STAT_ONCE_PER_FIELD = False

# fields to compare
```

(continues on next page)

(continued from previous page)

```
# Note: If FCST_VAR<n>_* is set, then a corresponding OBS_VAR<n>_* variable must be set
# To use one variables for both forecast and observation data, set BOTH_VAR<n>_* instead
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


# End of [config] section and start of [dir] section
[dir]
FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/met_test/new/point_stat_input/vvwind
OBS_POINT_STAT_INPUT_DIR = {INPUT_BASE}/met_test/new/point_stat_input/prepbufr

# directory containing climatology mean input to PointStat
# Not used in this example
POINT_STAT_CLIMO_MEAN_INPUT_DIR =

# directory containing climatology mean input to PointStat
# Not used in this example
POINT_STAT_CLIMO_STDEV_INPUT_DIR =


POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/point_stat_py_embed


# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to PointStat relative to FCST_POINT_STAT_INPUT_DIR
FCST_POINT_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to look for observation input to PointStat relative to OBS_POINT_STAT_INPUT_DIR
OBS_POINT_STAT_INPUT_TEMPLATE = gdas.{valid?fmt=%Y%m%d}.t{valid?fmt=%H}z.nc

# Template to look for climatology input to PointStat relative to POINT_STAT_CLIMO_MEAN_
↪INPUT_DIR
# Not used in this example
POINT_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

# Template to look for climatology input to PointStat relative to POINT_STAT_CLIMO_STDEV_
↪INPUT_DIR
# Not used in this example
POINT_STAT_CLIMO_STDEV_INPUT_TEMPLATE =
```

5.1.20.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVER-WRITTEN 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 68)

Note: See the *PointStat MET Configuration* (page 194) 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}

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

//
```

(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;
obs_summary    = NONE;
obs_perc_value = 50;
```

(continues on next page)

(continued from previous page)

```
//
// 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_GRID}
   ${METPLUS_MASK_POLY}
   ${METPLUS_MASK_SID}
   //llpnt =
   ${METPLUS_MASK_LLPNT}
}
```

(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}

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

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.20.2.7 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'),
```

(continues on next page)

(continued from previous page)

```
        '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'),
        '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:
```

(continues on next page)

(continued from previous page)

```
        # 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))
        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],
```

(continues on next page)

(continued from previous page)

```
    '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,
    }
}

print("Attributes: " + repr(attrs))
```

5.1.20.2.8 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.

5.1.20.2.9 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

5.1.20.2.10 Keywords

Note:

- PointStatToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.20.3 PointStat: Using Python Embedding for Point Observations

met_tool_wrapper/PointStat/PointStat_python_embedding_obs.conf

5.1.20.3.1 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.

5.1.20.3.2 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 560) section for more information.

Data Source: Unknown

5.1.20.3.3 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.

5.1.20.3.4 METplus Workflow

PointStat is the only tool called in this example. It processes the following run times:

Valid: 2007-03-30_0Z

5.1.20.3.5 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]

PROCESS_LIST = PointStat

###
# Time Info
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20070330
INIT_END = 20070330
INIT_INCREMENT = 1M

LEAD_SEQ = 36
```

(continues on next page)

(continued from previous page)

```
###
# File I/O
###

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/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
###

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
```

(continues on next page)

(continued from previous page)

```
FCST_VAR3_LEVELS = Z10
FCST_VAR3_THRESH = >=5
OBS_VAR3_NAME = VGRD
OBS_VAR3_LEVELS = Z10
OBS_VAR3_THRESH = >=5

###
# 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 =
#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 =
```

(continues on next page)

(continued from previous page)

```
#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 =

#POINT_STAT_MESSAGE_TYPE_GROUP_MAP =
```

5.1.20.3.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *PointStat MET Configuration* (page 194) 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}

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

//
```

(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;
obs_summary    = NONE;
obs_perc_value = 50;
```

(continues on next page)

(continued from previous page)

```
//
// 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_GRID}
   ${METPLUS_MASK_POLY}
   ${METPLUS_MASK_SID}
   //llpnt =
   ${METPLUS_MASK_LLPNT}
}
```

(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}

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

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.20.3.7 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.

5.1.20.3.8 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

5.1.20.3.9 Keywords

Note:

- PointStatToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.20.4 PointStat: Once Per Field

met_tool_wrapper/PointStat/PointStat_once_per_field.conf

5.1.20.4.1 Scientific Objective

Compare 3 hour forecast precipitation accumulations to observations of 3 hour precipitation accumulation. Generate statistics of the results.

5.1.20.4.2 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 571) section for more information.

Data Source: Unknown

5.1.20.4.3 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.

5.1.20.4.4 METplus Workflow

PointStat is the only tool called in this example. It processes the following run times:

Valid: 2007-03-30_0Z

5.1.20.4.5 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]

# List of applications to run - only PointStat for this case
PROCESS_LIST = PointStat

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
```

(continues on next page)

(continued from previous page)

```
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 20070330

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 20070330

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# Verbosity of MET output - overrides LOG_VERBOSITY for PointStat only
#LOG_POINT_STAT_VERBOSITY = 2

# Location of MET config file to pass to GridStat
# References PARM_BASE which is the location of the parm directory corresponding
# to the ush directory of the run_metplus.py script that is called
# or the value of the environment variable METPLUS_PARM_BASE if set
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
```

(continues on next page)

(continued from previous page)

```
# Time relative to each input file's valid time (in seconds if no units are specified) for␣
↪data within the file to be
# considered valid. Values are set in the 'obs_window' dictionary in the PointStat config␣
↪file
OBS_POINT_STAT_WINDOW_BEGIN = -5400
OBS_POINT_STAT_WINDOW_END = 5400

# Optional list of offsets to look for point observation data
POINT_STAT_OFFSETS = 0

# Model/fcst and obs name, e.g. GFS, NAM, GDAS, etc.
MODEL = WRF
OBTYPE =

#POINT_STAT_DESC =

# Regrid to specified grid.  Indicate NONE if no regridding, or the grid id
# (e.g. G212)
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}

# sets the -obs_valid_beg command line argument (optional)
# not used for this example
#POINT_STAT_OBS_VALID_BEG = {valid?fmt=%Y%m%d_%H}

# sets the -obs_valid_end command line argument (optional)
# not used for this example
#POINT_STAT_OBS_VALID_END = {valid?fmt=%Y%m%d_%H}

# Verification Masking regions
# Indicate which grid and polygon masking region, if applicable
POINT_STAT_GRID = DTC165, DTC166

# List of full path to poly masking files.  NOTE: Only short lists of poly
# files work (those that fit on one line), a long list will result in an
# environment variable that is too long, resulting in an error.  For long
# lists of poly masking files (i.e. all the mask files in the NCEP_mask
# directory), define these in the MET point_stat configuration file.
POINT_STAT_POLY = MET_BASE/poly/LMV.poly
POINT_STAT_STATION_ID =
```

(continues on next page)

(continued from previous page)

```
# Message types, if all message types are to be returned, leave this empty,
# otherwise indicate the message types of interest.
POINT_STAT_MESSAGE_TYPE = ADPUPA, ADPSFC
# Variables and levels as specified in the field dictionary of the MET
# point_stat configuration file. Specify as FCST_VARn_NAME, FCST_VARn_LEVELS,
# (optional) FCST_VARn_OPTION

# set to True to run PointStat once for each name/level combination
# set to False to run PointStat once per run time including all fields
POINT_STAT_ONCE_PER_FIELD = True

# fields to compare
# Note: If FCST_VAR<n>_* is set, then a corresponding OBS_VAR<n>_* variable must be set
# To use one variables for both forecast and observation data, set BOTH_VAR<n>_* instead
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


# End of [config] section and start of [dir] section
[dir]
FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst
OBS_POINT_STAT_INPUT_DIR = {INPUT_BASE}/met_test/out/pb2nc

# directory containing climatology mean input to PointStat
# Not used in this example
POINT_STAT_CLIMO_MEAN_INPUT_DIR =
```

(continues on next page)

(continued from previous page)

```
# directory containing climatology mean input to PointStat
# Not used in this example
POINT_STAT_CLIMO_STDEV_INPUT_DIR =


POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/point_stat


# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to PointStat relative to FCST_POINT_STAT_INPUT_DIR
FCST_POINT_STAT_INPUT_TEMPLATE = {valid?fmt=%Y%m%d%H}/nam.t00z.awip1236.tm00.{valid?fmt=%Y%m
↪%d}.grb

# Template to look for observation input to PointStat relative to OBS_POINT_STAT_INPUT_DIR
OBS_POINT_STAT_INPUT_TEMPLATE = sample_pb.nc

# Template to look for climatology input to PointStat relative to POINT_STAT_CLIMO_MEAN_
↪INPUT_DIR
# Not used in this example
POINT_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

# Template to look for climatology input to PointStat relative to POINT_STAT_CLIMO_STDEV_
↪INPUT_DIR
# Not used in this example
POINT_STAT_CLIMO_STDEV_INPUT_TEMPLATE =
```

5.1.20.4.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *PointStat MET Configuration* (page 194) 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_GRID}
   ${METPLUS_MASK_POLY}
   ${METPLUS_MASK_SID}
   //llpnt =
   ${METPLUS_MASK_LLPNT}
}

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

//
// 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}

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

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.20.4.7 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.

5.1.20.4.8 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

5.1.20.4.9 Keywords

Note:

- PointStatToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.21 PyEmbedIngest

5.1.21.1 PyEmbedIngest: Multiple Fields in One File

met_tool_wrapper/PyEmbedIngest/PyEmbedIngest_multi_field_one_file.conf

5.1.21.1.1 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.

5.1.21.1.2 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 575) section for more information.

5.1.21.1.3 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.

5.1.21.1.4 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.

5.1.21.1.5 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

```
# PyEmbedIngest wrapper example

[config]
# Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = times

# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 2013022712

# End time for METplus run
VALID_END = 2013022712

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 21600

# list of forecast leads to process
LEAD_SEQ = 0

# List of applications to run
```

(continues on next page)

(continued from previous page)

```
PROCESS_LIST = PyEmbedIngest


# 1st INGEST INSTANCE
# python script with optional arguments to run for 1st ingest instance
# this ingest instance runs 1 scripts to generate 1 field
PY_EMBED_INGEST_1_SCRIPT_1 = {INPUT_BASE}/met_test/scripts/python/read_ascii_numpy.py {INPUT_
↪BASE}/met_test/data/python/fcst.txt FCST

# type of python input to expect for 1st ingest instance
# valid options: NUMPY, XARRAY
PY_EMBED_INGEST_1_TYPE = NUMPY

# output grid for 1st ingest instance. Can be a grid definition or file path
PY_EMBED_INGEST_1_OUTPUT_GRID = G130


# 2nd INGEST INSTANCE
# python script with optional arguments to run for 2nd ingest instance
# this ingest instance runs 2 scripts to generate 2 fields
PY_EMBED_INGEST_2_SCRIPT_1 = {INPUT_BASE}/met_test/scripts/python/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 = {INPUT_BASE}/met_test/scripts/python/read_ascii_numpy.py {INPUT_
↪BASE}/met_test/data/python/obs.txt OBS
PY_EMBED_INGEST_2_OUTPUT_FIELD_NAME_2 = Observation

# type of python input to expect for 2nd ingest instance
# valid options: NUMPY, XARRAY
PY_EMBED_INGEST_2_TYPE = NUMPY

# output grid for 2nd ingest instance. Can be a grid definition or file path
PY_EMBED_INGEST_2_OUTPUT_GRID = G130

[dir]
# output directory for 1st ingest instance
# in this example, the full output path is set in PY_EMBED_INGEST_1_OUTPUT_TEMPLATE
PY_EMBED_INGEST_1_OUTPUT_DIR =

# output directory for 2nd ingest instance
# in this example, the full output path is set in PY_EMBED_INGEST_2_OUTPUT_TEMPLATE
PY_EMBED_INGEST_2_OUTPUT_DIR =

[filename_templates]
```

(continues on next page)

(continued from previous page)

```
# output template to use for 1st ingest instance
# can optionally use [dir] PY_EMBED_INGEST_1_OUTPUT_DIR with this value
PY_EMBED_INGEST_1_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/PyEmbedIngest/test.nc

# output template to use for 2nd ingest instance
# can optionally use [dir] PY_EMBED_INGEST_2_OUTPUT_DIR with this value
PY_EMBED_INGEST_2_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/PyEmbedIngest/regrid_data_
↪plane.nc
```

5.1.21.1.6 MET Configuration

None. RegridDataPlane does not use configuration files.

5.1.21.1.7 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

5.1.21.1.8 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.

5.1.21.1.9 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

5.1.21.1.10 Keywords

Note:

- PyEmbedIngestToolUseCase
- PythonEmbeddingFileUseCase
- RegridDataPlaneToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.21.2 PyEmbedIngest: Basic Use Case

met_tool_wrapper/PyEmbedIngest/PyEmbedIngest.conf

5.1.21.2.1 Scientific Objective

None. Simply converting file formats so data can be read by the MET tools.

5.1.21.2.2 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 579) section for more information.

5.1.21.2.3 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.

5.1.21.2.4 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.

5.1.21.2.5 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

```
# PyEmbedIngest wrapper example

[config]
# Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = times
```

(continues on next page)

(continued from previous page)

```
# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 2013022712

# End time for METplus run
VALID_END = 2013022712

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 21600

# list of forecast leads to process
LEAD_SEQ = 0

# List of applications to run
PROCESS_LIST = PyEmbedIngest


# 1st INGEST INSTANCE
# python script with optional arguments to run for 1st ingest instance
PY_EMBED_INGEST_1_SCRIPT = {INPUT_BASE}/met_test/scripts/python/read_ascii_numpy.py {INPUT_
↪BASE}/met_test/data/python/obs.txt OBS

# type of python input to expect for 1st ingest instance
# valid options: NUMPY, XARRAY
PY_EMBED_INGEST_1_TYPE = NUMPY

# output grid for 1st ingest instance. Can be a grid definition or file path
PY_EMBED_INGEST_1_OUTPUT_GRID = G130


# 2nd INGEST INSTANCE
# python script with optional arguments to run for 2nd ingest instance
PY_EMBED_INGEST_2_SCRIPT = {INPUT_BASE}/met_test/scripts/python/read_ascii_numpy.py {INPUT_
↪BASE}/met_test/data/python/fcst.txt FCST

# type of python input to expect for 2nd ingest instance
# valid options: NUMPY, XARRAY
PY_EMBED_INGEST_2_TYPE = NUMPY

# output grid for 2nd ingest instance. Can be a grid definition or file path
```

(continues on next page)

(continued from previous page)

```
PY_EMBED_INGEST_2_OUTPUT_GRID = G130

[dir]
# output directory for 1st ingest instance
# in this example, the full output path is set in PY_EMBED_INGEST_1_OUTPUT_TEMPLATE
PY_EMBED_INGEST_1_OUTPUT_DIR =

# output directory for 2nd ingest instance
# in this example, the full output path is set in PY_EMBED_INGEST_2_OUTPUT_TEMPLATE
PY_EMBED_INGEST_2_OUTPUT_DIR =

[filename_templates]
# output template to use for 1st ingest instance
# can optionally use [dir] PY_EMBED_INGEST_1_OUTPUT_DIR with this value
PY_EMBED_INGEST_1_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/PyEmbedIngest/test.nc

# output template to use for 2nd ingest instance
# can optionally use [dir] PY_EMBED_INGEST_2_OUTPUT_DIR with this value
PY_EMBED_INGEST_2_OUTPUT_TEMPLATE = {OUTPUT_BASE}/met_tool_wrapper/PyEmbedIngest/regrid_data_
↪plane.nc
```

5.1.21.2.6 MET Configuration

None. RegridDataPlane does not use configuration files.

5.1.21.2.7 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

5.1.21.2.8 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.

5.1.21.2.9 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

5.1.21.2.10 Keywords

Note:

- PyEmbedIngestToolUseCase
- PythonEmbeddingFileUseCase
- RegridDataPlaneToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.22 RegridDataPlane

5.1.22.1 RegridDataPlane: Process all fields

met_tool_wrapper/RegridDataPlane/RegridDataPlane_multi_field_one _file.conf

5.1.22.1.1 Scientific Objective

Simply regridding data to match a desired grid domain for comparisons. Process all fields in a single call to RegridDataPlane

5.1.22.1.2 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 584) section for more information.

5.1.22.1.3 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.

5.1.22.1.4 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.

5.1.22.1.5 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]
# List of applications to run - only RegridDataPlane for this case
PROCESS_LIST = RegridDataPlane

# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG=2005080700

# End time for METplus run - must match INIT_TIME_FMT
INIT_END=2005080700

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 3H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
```

(continues on next page)

(continued from previous page)

```
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# If True, run regrid_data_plane on observation data
OBS_REGRID_DATA_PLANE_RUN = True

# name of input field to process
# if unset, OBS_VAR1_NAME will be used
OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = APCP
OBS_REGRID_DATA_PLANE_VAR2_INPUT_FIELD_NAME = TMP

# level of input field to process
# if unset, OBS_VAR1_LEVELS will be used
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = L0
OBS_REGRID_DATA_PLANE_VAR2_INPUT_LEVEL = L0

# name of output field to create
# if unset, OBS_VAR1_NAME and OBS_VAR1_LEVELS will be combined to generate an output field␣
↪name
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = APCP_01

# 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

# If running a MET tool comparison tool after RegridDataPlane, one can instead set OBS_VAR1_
↪[NAME/LEVELS] to
# a value that corresponds to the desired name/level to use in the comparison
# this value will be used to determine output name/level to pass to RegridDataPlane as well
#OBS_VAR1_NAME = APCP
#OBS_VAR1_LEVELS = A01

# Name to identify model data in output
MODEL = QPF

# Name to identify observation data in output
OBTYPE = QPE

# Used by regrid_data_plane to remap data
REGRID_DATA_PLANE_VERIF_GRID={INPUT_BASE}/met_test/data/sample_obs/ST2ml/ST2ml2005080703.Grb_
↪G212
```

(continues on next page)

(continued from previous page)

```
# method to run regrid_data_plane, not setting this will default to NEAREST
REGRID_DATA_PLANE_METHOD = BUDGET

# regridding width used in regrid_data_plane, not setting this will default to 1
REGRID_DATA_PLANE_WIDTH = 2

# Gaussian filter DX value to add as command line argument - not added if unset or blank
REGRID_DATA_PLANE_GAUSSIAN_DX =

# Gaussian filter radius value to add as command line argument - not added if unset or blank
REGRID_DATA_PLANE_GAUSSIAN_RADIUS =

# End of [config] section and start of [dir] section
[dir]
# directory containing observation input to RegridDataPlane
OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst

# directory to write observation output from RegridDataPlane
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/RegridDataPlane/multi_
↪field_one_file

# End of [dir] section and start of [filename_templates] section
[filename_templates]
# template to use to read input data and write output data for RegridDataPlane
# if different names for input and output are desired, set OBS_REGRID_DATA_PLANE_INPUT_
↪TEMPLATE
# and OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE instead
OBS_REGRID_DATA_PLANE_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%2H}.tm00_G212
```

5.1.22.1.6 MET Configuration

None. RegridDataPlane does not use configuration files.

5.1.22.1.7 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.

5.1.22.1.8 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

5.1.22.1.9 Keywords

Note:

- RegridDataPlaneToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.22.2 RegridDataPlane: Using Python Embedding

met_tool_wrapper/RegridDataPlane/RegridDataPlane_python _embedding.conf

5.1.22.2.1 Scientific Objective

None. Simply regridding data to match a desired grid domain for comparisons.

5.1.22.2.2 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 588) section for more information.

5.1.22.2.3 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.

5.1.22.2.4 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.

5.1.22.2.5 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]
# List of applications to run
PROCESS_LIST = RegridDataPlane

# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = INIT

# Format of INIT_BEG and INT_END
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run
INIT_BEG=2005080700

# End time for METplus run
INIT_END=2005080700

# Increment between METplus runs in seconds. Must be >= 60
INIT_INCREMENT=43200

# List of forecast leads to process
LEAD_SEQ = 3

# Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = times

# run regrid_data_plane on observation data
OBS_REGRID_DATA_PLANE_RUN = True

# List of variables to compare
OBS_VAR1_NAME = {INPUT_BASE}/met_test/scripts/python/read_ascii_numpy.py {INPUT_BASE}/met_
↪test/data/python/obs.txt OBS

OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = OBS

# 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 = True
```

(continues on next page)

(continued from previous page)

```
# Name to identify model data in output
MODEL = FCST

# Name to identify observation data in output
OBTYPE = OBS

# Used by regrid_data_plane to remap data
REGRID_DATA_PLANE_VERIF_GRID = G130

# method to run regrid_data_plane, not setting this will default to NEAREST
REGRID_DATA_PLANE_METHOD = BUDGET

# regridding width used in regrid_data_plane, not setting this will default to 1
REGRID_DATA_PLANE_WIDTH = 2

[dir]
OBS_REGRID_DATA_PLANE_INPUT_DIR =
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/RegridDataPlane/regrid_py

[filename_templates]
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = PYTHON_NUMPY
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = numpy_data.nc
```

5.1.22.2.6 MET Configuration

None. RegridDataPlane does not use configuration files.

5.1.22.2.7 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

5.1.22.2.8 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.

5.1.22.2.9 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

5.1.22.2.10 Keywords

Note:

- RegridDataPlaneToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.22.3 RegridDataPlane: Run once per field

met_tool_wrapper/RegridDataPlane/RegridDataPlane_multi_field _multi_file.conf

5.1.22.3.1 Scientific Objective

Simply regridding data to match a desired grid domain for comparisons. Process each field separately and write a file for each.

5.1.22.3.2 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 594) section for more information.

5.1.22.3.3 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.

5.1.22.3.4 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.

5.1.22.3.5 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]
# List of applications to run - only RegridDataPlane for this case
PROCESS_LIST = RegridDataPlane

# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG=2005080700

# End time for METplus run - must match INIT_TIME_FMT
INIT_END=2005080700

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 3H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
```

(continues on next page)

(continued from previous page)

```
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# If True, run regrid_data_plane on observation data
OBS_REGRID_DATA_PLANE_RUN = True

# name of input field to process
# if unset, OBS_VAR1_NAME will be used
OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = APCP
OBS_REGRID_DATA_PLANE_VAR2_INPUT_FIELD_NAME = TMP

# level of input field to process
# if unset, OBS_VAR1_LEVELS will be used
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = L0
OBS_REGRID_DATA_PLANE_VAR2_INPUT_LEVEL = L0

# name of output field to create
# if unset, OBS_VAR1_NAME and OBS_VAR1_LEVELS will be combined to generate an output field␣
↪name
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = APCP_01

# 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 = True

# If running a MET tool comparison tool after RegridDataPlane, one can instead set OBS_VAR1_
↪[NAME/LEVELS] to
# a value that corresponds to the desired name/level to use in the comparison
# this value will be used to determine output name/level to pass to RegridDataPlane as well
#OBS_VAR1_NAME = APCP
#OBS_VAR1_LEVELS = A01

# Name to identify model data in output
MODEL = QPF

# Name to identify observation data in output
OBTYPE = QPE

# Used by regrid_data_plane to remap data
REGRID_DATA_PLANE_VERIF_GRID={INPUT_BASE}/met_test/data/sample_obs/ST2ml/ST2ml2005080703.Grb_
↪G212

# method to run regrid_data_plane, not setting this will default to NEAREST
```

(continues on next page)

(continued from previous page)

```
REGRID_DATA_PLANE_METHOD = BUDGET

# regridding width used in regrid_data_plane, not setting this will default to 1
REGRID_DATA_PLANE_WIDTH = 2

# Gaussian filter DX value to add as command line argument - not added if unset or blank
REGRID_DATA_PLANE_GAUSSIAN_DX =

# Gaussian filter radius value to add as command line argument - not added if unset or blank
REGRID_DATA_PLANE_GAUSSIAN_RADIUS =

# End of [config] section and start of [dir] section
[dir]
# directory containing observation input to RegridDataPlane
OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_fcst

# directory to write observation output from RegridDataPlane
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/RegridDataPlane/multi_
↪field_multi_file


# End of [dir] section and start of [filename_templates] section
[filename_templates]
# template to use to read input data and write output data for RegridDataPlane
# if different names for input and output are desired, set OBS_REGRID_DATA_PLANE_INPUT_
↪TEMPLATE
# and OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE instead
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_ruc13_{lead?fmt=%2H}.tm00_
↪G212
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/wrfprs_{obs_name}_{lead?fmt=%2H}.
↪tm00_G212
```

5.1.22.3.6 MET Configuration

None. RegridDataPlane does not use configuration files.

5.1.22.3.7 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.

5.1.22.3.8 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

5.1.22.3.9 Keywords

Note:

- RegridDataPlaneToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.22.4 RegridDataPlane: Basic Use Case

met_tool_wrapper/RegridDataPlane/RegridDataPlane.conf

5.1.22.4.1 Scientific Objective

Simply regridding data to match a desired grid domain for comparisons.

5.1.22.4.2 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 599) section for more information.

5.1.22.4.3 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.

5.1.22.4.4 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.

5.1.22.4.5 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]
# List of applications to run - only RegridDataPlane for this case
PROCESS_LIST = RegridDataPlane

# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG=2005080700

# End time for METplus run - must match INIT_TIME_FMT
INIT_END=2005080700
```

(continues on next page)

(continued from previous page)

```
# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 3H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# If True, run regrid_data_plane on observation data
OBS_REGRID_DATA_PLANE_RUN = True

# name of input field to process
# if unset, OBS_VAR1_NAME will be used
OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = APCP

# level of input field to process
# if unset, OBS_VAR1_LEVELS will be used
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = A01

# name of output field to create
# if unset, OBS_VAR1_NAME and OBS_VAR1_LEVELS will be combined to generate an output field␣
↪name
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = APCP_01

# If running a MET tool comparison tool after RegridDataPlane, one can instead set OBS_VAR1_
↪[NAME/LEVELS] to
# a value that corresponds to the desired name/level to use in the comparison
# this value will be used to determine output name/level to pass to RegridDataPlane as well
#OBS_VAR1_NAME = APCP
#OBS_VAR1_LEVELS = A01

# 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 = True
```

(continues on next page)

(continued from previous page)

```
# Name to identify model data in output
MODEL = QPF

# Name to identify observation data in output
OBTYPE = QPE

# Used by regrid_data_plane to remap data
REGRID_DATA_PLANE_VERIF_GRID={INPUT_BASE}/met_test/data/sample_fcst/2005080700/wrfprs_ruc13_
↪12.tm00_G212

# method to run regrid_data_plane, not setting this will default to NEAREST
REGRID_DATA_PLANE_METHOD = BUDGET

# regridding width used in regrid_data_plane, not setting this will default to 1
REGRID_DATA_PLANE_WIDTH = 2

# Gaussian filter DX value to add as command line argument - not added if unset or blank
REGRID_DATA_PLANE_GAUSSIAN_DX =

# Gaussian filter radius value to add as command line argument - not added if unset or blank
REGRID_DATA_PLANE_GAUSSIAN_RADIUS =

# End of [config] section and start of [dir] section
[dir]
# directory containing observation input to RegridDataPlane
OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/met_test/data/sample_obs/ST2ml

# directory to write observation output from RegridDataPlane
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/regrid_data_plane


# End of [dir] section and start of [filename_templates] section
[filename_templates]
# template to use to read input data and write output data for RegridDataPlane
# if different names for input and output are desired, set OBS_REGRID_DATA_PLANE_INPUT_
↪TEMPLATE
# and OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE instead
OBS_REGRID_DATA_PLANE_TEMPLATE = ST2ml{valid?fmt=%Y%m%d%H}.Grb_G212
```

5.1.22.4.6 MET Configuration

None. RegridDataPlane does not use configuration files.

5.1.22.4.7 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.

5.1.22.4.8 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

5.1.22.4.9 Keywords

Note:

- RegridDataPlaneToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.23 SeriesAnalysis

5.1.23.1 SeriesAnalysis: Using Python Embedding

met_tool_wrapper/SeriesAnalysis/SeriesAnalysis_python _embedding.conf

5.1.23.1.1 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.

5.1.23.1.2 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 606) section for more information.

5.1.23.1.3 METplus Components

This use case utilizes the METplus SeriesAnalysis wrapper to search for files as determined by the Python script.

5.1.23.1.4 METplus Workflow

SeriesAnalysis is the only tool called in this example. It processes simple text files with no timing or meteorological information to demonstrate how SeriesAnalysis can be run utilizing Python Embedding.

5.1.23.1.5 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

```
# SeriesAnalysis using Python Embedding METplus Configuration

[config]

PROCESS_LIST = SeriesAnalysis

LOOP_ORDER = processes

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 =

SERIES_ANALYSIS_DESC =

SERIES_ANALYSIS_CAT_THRESH =

SERIES_ANALYSIS_VLD_THRESH =

SERIES_ANALYSIS_BLOCK_SIZE =
```

(continues on next page)

(continued from previous page)

```
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_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False

#LOG_SERIES_ANALYSIS_VERBOSITY = 2

SERIES_ANALYSIS_IS_PAIRED = False

SERIES_ANALYSIS_CONFIG_FILE = {CONFIG_DIR}/SeriesAnalysisConfig_wrapped

SERIES_ANALYSIS_STAT_LIST = TOTAL, RMSE, FBAR, OBAR

MODEL = PYTHON

OBTYPE = ANALYS

FCST_SERIES_ANALYSIS_INPUT_DATATYPE = PYTHON_NUMPY

OBS_SERIES_ANALYSIS_INPUT_DATATYPE = PYTHON_NUMPY

FCST_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/read_ascii_numpy.py MET_PYTHON_INPUT_ARG␣
↪FCST

OBS_VAR1_NAME = {MET_INSTALL_DIR}/share/met/python/read_ascii_numpy.py MET_PYTHON_INPUT_ARG␣
↪OBS

[dir]

CONFIG_DIR={PARM_BASE}/met_config

FCST_SERIES_ANALYSIS_INPUT_DIR = {INPUT_BASE}/met_test/data/python

OBS_SERIES_ANALYSIS_INPUT_DIR = {INPUT_BASE}/met_test/data/python

SERIES_ANALYSIS_CLIMO_MEAN_INPUT_DIR =
```

(continues on next page)

(continued from previous page)

```
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR =

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/SeriesAnalysis

[filename_templates]

FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = fcst.txt, fcst.txt

OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = obs.txt, obs.txt

SERIES_ANALYSIS_OUTPUT_TEMPLATE = python_sa.nc

SERIES_ANALYSIS_CLIMO_MEAN_INPUT_TEMPLATE =

SERIES_ANALYSIS_CLIMO_STDEV_INPUT_TEMPLATE =
```

5.1.23.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *SeriesAnalysis MET Configuration* (page 207) 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}
```

(continues on next page)

(continued from previous page)

```
//
// 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 = {
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

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

(continues on next page)

(continued from previous page)

```
//
// 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 = {
   grid = "";
   poly = "";
}

//
// 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}
```

(continues on next page)

(continued from previous page)

```
//
// 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}
```

5.1.23.1.7 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

5.1.23.1.8 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.

5.1.23.1.9 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

5.1.23.1.10 Keywords

Note:

- SeriesAnalysisUseCase
- PythonEmbeddingFileUseCase
- DiagnosticsUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.23.2 SeriesAnalysis: Basic Use Case

met_tool_wrapper/SeriesAnalysis/SeriesAnalysis.conf

5.1.23.2.1 Scientific Objective

Compare forecasts for 3-hour precipitation accumulations to observed 3-hour accumulation. These comparisons are made through generating statistics of the results.

5.1.23.2.2 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 615) section for more information.

5.1.23.2.3 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.

5.1.23.2.4 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

5.1.23.2.5 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]

PROCESS_LIST = SeriesAnalysis

###
# Time Info
###

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_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False

SERIES_ANALYSIS_CUSTOM_LOOP_LIST =

###
# File I/O
###

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 =
```

(continues on next page)

(continued from previous page)

```
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR =
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_TEMPLATE =


###
# Field Info
###

MODEL = WRF
OBTYPE = MC_PCP

#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
###

#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 =
```

(continues on next page)

(continued from previous page)

```
#SERIES_ANALYSIS_REGRID_SHAPE =

#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 =
#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 =
```

(continues on next page)

(continued from previous page)

```
#SERIES_ANALYSIS_OUTPUT_STATS_PSTD =
#SERIES_ANALYSIS_OUTPUT_STATS_PJC =
#SERIES_ANALYSIS_OUTPUT_STATS_PRC =


###
# Plotting
###

SERIES_ANALYSIS_GENERATE_PLOTS = no

PLOT_DATA_PLANE_TITLE =

SERIES_ANALYSIS_GENERATE_ANIMATIONS = no
```

5.1.23.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *SeriesAnalysis MET Configuration* (page 207) 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}

//
```

(continues on next page)

(continued from previous page)

```
// 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 = {
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

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

(continues on next page)

(continued from previous page)

```
//
// 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 = {
   grid = "";
   poly = "";
}

//
// 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}
```

(continues on next page)

(continued from previous page)

```
//
// 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}
```

5.1.23.2.7 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.

5.1.23.2.8 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

5.1.23.2.9 Keywords

Note:

- SeriesAnalysisUseCase
- DiagnosticsUseCase
- ListExpansionFeatureUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.24 StatAnalysis

5.1.24.1 StatAnalysis: Basic Use Case

met_tool_wrapper/StatAnalysis/StatAnalysis.conf

5.1.24.1.1 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.

5.1.24.1.2 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 624) section for more information.
Data Source: WRF

5.1.24.1.3 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.

5.1.24.1.4 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

5.1.24.1.5 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

```
# StatAnalysis METplus Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# List of applications to run - only StatAnalysis for this case
PROCESS_LIST = StatAnalysis

# time looping - 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
LOOP_BY = VALID

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H


# Start time for METplus run - must match INIT_TIME_FMT
VALID_BEG=2005080700

# End time for METplus run - must match INIT_TIME_FMT
VALID_END=2005080700

# Increment between METplus runs (in seconds if no units are specified)
```

(continues on next page)

(continued from previous page)

```
#  Must be >= 60 seconds
VALID_INCREMENT = 12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 12

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for StatAnalysis only
#LOG_STAT_ANALYSIS_VERBOSITY = 2

# Models to process
# MODELn is the model name to filter for in
#           stat files [required]
# MODELn_OBTYPE is the observation name
#               to filter for the .stat files
#               [required]
# MODELn_STAT_ANALYSIS_LOOKIN_DIR is the directory to search for
#                                  the .stat files in, wildcards (*)
#                                  are okay to search for multiple
#                                  directories and templates like
#                                  {valid?fmt=%Y%m%d%H%M%S} [required]
# MODELn_REFERENCE_NAME is a reference name for MODELn, defaults to
#                        MODELn, it can be used in the file template names
#                        [optional]
MODEL1 = WRF
MODEL1_OBTYPE = ANALYS


# Location of MET config file to pass to StatAnalysis
# References CONFIG_DIR from the [dir] section
STAT_ANALYSIS_CONFIG_FILE = {CONFIG_DIR}/STATAnalysisConfig_wrapped

#STAT_ANALYSIS_HSS_EC_VALUE =
```

(continues on next page)

(continued from previous page)

```
# stat_analysis job info
STAT_ANALYSIS_JOB_NAME = filter
# 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 = -dump_row [dump_row_file]

# Optional variables for further filtering
# can be blank, single, or multiple values
# if more than one use comma separated list
#
# (FCST)(OBS)_(VALID)(INIT)_HOUR_LIST: HH format (ex. 00, 06, 12)
# (FCST)(OBS)_LEAD_LIST: HH[H][MMSS] format (ex. 00, 06, 120)
MODEL_LIST = {MODEL1}
DESC_LIST =
FCST_LEAD_LIST =
OBS_LEAD_LIST =
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 =
# how to treat items listed in above _LIST variables
# GROUP_LIST_ITEMS: items listed in a given _LIST variable
#                   will be grouped together
# LOOP_LIST_ITEMS:  items listed in a give _LIST variable
#                   will be looped over
# if not listed METplus will treat the list as a group
GROUP_LIST_ITEMS = FCST_INIT_HOUR_LIST
LOOP_LIST_ITEMS = FCST_VALID_HOUR_LIST, MODEL_LIST
```

(continues on next page)

(continued from previous page)

```
MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {INPUT_BASE}/met_test/out/grid_stat

# Output data directory
STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/stat_analysis

STAT_ANALYSIS_OUTPUT_TEMPLATE = job.out

# location of configuration files used by MET applications
CONFIG_DIR = {PARM_BASE}/met_config


# Optional settings to create templated directory and file name information
# to save files as stat_analysis output as, this is appended to STAT_ANALYSIS_OUTPUT_DIR
# if no template is provided a default filename set in the code will be used
# Use:
# string templates can be set for all the lists being looped over, just
# use and a lower case version of the list, ex. {fcst_valid_hour?fmt=%H}
# or {fcst_var?fmt=%s}
# For looping over models:
# can set MODELn_STAT_ANALYSIS_[DUMP_ROW/OUT_STAT]_TEMPLATE for individual models
# or STAT_ANALYSIS_[DUMP_ROW/OUT_STAT] with {model?fmt=%s}
MODEL1_STAT_ANALYSIS_DUMP_ROW_TEMPLATE = {fcst_valid_hour?fmt=%H}Z/{MODEL1}/{MODEL1}_{valid?
↪fmt=%Y%m%d%H}.stat
```

5.1.24.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *StatAnalysis MET Configuration* (page 218) 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}
```

(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}
```

5.1.24.1.7 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.

5.1.24.1.8 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

5.1.24.1.9 Keywords

Note:

- StatAnalysisToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.24.2 StatAnalysis: Using Python Embedding

met_tool_wrapper/StatAnalysis/StatAnalysis_python_embedding.conf

5.1.24.2.1 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.

5.1.24.2.2 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 633) section for more information.

Data Source: WRF

5.1.24.2.3 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.

5.1.24.2.4 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

5.1.24.2.5 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

```
# StatAnalysis METplus Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# List of applications to run - only StatAnalysis for this case
PROCESS_LIST = StatAnalysis

# time looping - 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
LOOP_BY = VALID

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
```

(continues on next page)

(continued from previous page)

```
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H


# Start time for METplus run - must match INIT_TIME_FMT
VALID_BEG=2005080700

# End time for METplus run - must match INIT_TIME_FMT
VALID_END=2005080700

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 12

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for StatAnalysis only
#LOG_STAT_ANALYSIS_VERBOSITY = 2

# Models to process
# MODELn is the model name to filter for in
#           stat files [required]
# MODELn_OBTYPE is the observation name
#               to filter for the .stat files
#               [required]
# MODELn_STAT_ANALYSIS_LOOKIN_DIR is the directory to search for
#                                 the .stat files in, wildcards (*)
#                                 are okay to search for multiple
#                                 directories and templates like
#                                 {valid?fmt=%Y%m%d%H%M%S} [required]
# MODELn_REFERENCE_NAME is a reference name for MODELn, defaults to
```

(continues on next page)

(continued from previous page)

```
#                        MODELn, it can be used in the file template names
#                        [optional]
MODEL1 = WRF
MODEL1_OBTYPE = ADPSFC


# Location of MET config file to pass to StatAnalysis
# References CONFIG_DIR from the [dir] section
STAT_ANALYSIS_CONFIG_FILE = {CONFIG_DIR}/STATAnalysisConfig_wrapped

# 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 sl1l2 -by FCST_VAR -out_stat [out_stat_file]

# Optional variables for further filtering
# can be blank, single, or multiple values
# if more than one use comma separated list
#
# (FCST)(OBS)_(VALID)(INIT)_HOUR_LIST: HH format (ex. 00, 06, 12)
# (FCST)(OBS)_LEAD_LIST: HH[H][MMSS] format (ex. 00, 06, 120)
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 =
LINE_TYPE_LIST = MPR
```

(continues on next page)

(continued from previous page)

```
# how to treat items listed in above _LIST variables
# GROUP_LIST_ITEMS: items listed in a given _LIST variable
#                   will be grouped together
# LOOP_LIST_ITEMS:  items listed in a give _LIST variable
#                   will be looped over
# if not listed METplus will treat the list as a group
GROUP_LIST_ITEMS = FCST_INIT_HOUR_LIST
LOOP_LIST_ITEMS = FCST_VALID_HOUR_LIST, MODEL_LIST


# End of [config] section and start of [dir] section
[dir]

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = python {INPUT_BASE}/met_test/scripts/python/read_ascii_mpr.
↪py {INPUT_BASE}/met_test/new/point_stat_120000L_20050807_120000V.stat

# Output data directory
STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/StatAnalysis_python_embedding

# location of configuration files used by MET applications
CONFIG_DIR = {PARM_BASE}/met_config


# End of [dir] section and start of [filename_templates] section
[filename_templates]
# Optional settings to create templated directory and file name information
# to save files as stat_analysis output as, this is appended to STAT_ANALYSIS_OUTPUT_DIR
# if no template is provided a default filename set in the code will be used
# Use:
# string templates can be set for all the lists being looped over, just
# use and a lower case version of the list, ex. {fcst_valid_hour?fmt=%H}
# or {fcst_var?fmt=%s}
# For looping over models:
# can set MODELn_STAT_ANALYSIS_[DUMP_ROW/OUT_STAT]_TEMPLATE for individual models
# or STAT_ANALYSIS_[DUMP_ROW/OUT_STAT] with {model?fmt=%s}
MODEL1_STAT_ANALYSIS_DUMP_ROW_TEMPLATE = stat_analysis_python_AGGR_MPR_to_SL1L2.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
```

5.1.24.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *StatAnalysis MET Configuration* (page 218) 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}
```

5.1.24.2.7 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

5.1.24.2.8 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.

5.1.24.2.9 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

5.1.24.2.10 Keywords

Note:

- StatAnalysisToolUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.25 TCGen

5.1.25.1 TCGen: Basic Use Case

met_tool_wrapper/TCGen/TCGen.conf

5.1.25.1.1 Scientific Objective

The TC-Gen tool provides verification of tropical cyclone genesis forecasts in ATCF file format.

5.1.25.1.2 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 646) section for more information.

5.1.25.1.3 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.

5.1.25.1.4 METplus Workflow

TCGen is the only tool called in this example. It processes the following run times:

Init: 2016

5.1.25.1.5 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]

# Looping by times: steps through each 'task' in the PROCESS_LIST for each
# defined time, and repeats until all times have been evaluated.
LOOP_ORDER = times

# 'Tasks' to be run
PROCESS_LIST = TCGen

LOOP_BY = INIT

# The init time
INIT_TIME_FMT = %Y
INIT_BEG = 2016


LOG_TC_GEN_VERBOSITY = 2

# optional list of strings to loop over and call the tool multiple times
# value of each item can be referenced in filename templates with {custom?fmt=%s}
TC_GEN_CUSTOM_LOOP_LIST =


# I/O Configurations

# Location of input data directory for track data
TC_GEN_TRACK_INPUT_DIR = {INPUT_BASE}/met_test/tc_data/genesis/atcf
TC_GEN_TRACK_INPUT_TEMPLATE = {init?fmt=%Y}/*.dat

# Location of input data directory for genesis data
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 =


# directory to write output files generated by tc_gen
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}


# MET Configurations

TC_GEN_CONFIG_FILE = {PARM_BASE}/met_config/TCGenConfig_wrapped

# The following variables set values in the MET configuration file used by this example
# Leaving these values commented will use the value found in the default MET configuration␣
↪file
# See the MET documentation for this tool for more information on the settings

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";
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;
```

(continues on next page)

(continued from previous page)

```
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 =

# sets METPLUS_LEAD in the wrapped MET config file
LEAD_SEQ =

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

TC_GEN_DEV_HIT_RADIUS = 500

TC_GEN_DEV_HIT_WINDOW_BEGIN = -24
```

(continues on next page)

(continued from previous page)

```
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
```

5.1.25.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *TCGen MET Configuration* (page 227) 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}
```

5.1.25.1.7 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.

5.1.25.1.8 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

5.1.25.1.9 Keywords

Note:

- TCGenToolUseCase
- DTCOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.26 TCMPRPlotter

5.1.26.1 TCMPRPlotter: Basic Use Case

met_tool_wrapper/TCMPRPlotter/TCMPRPlotter.conf

5.1.26.1.1 Scientific Objective

Generate plots of tropical cyclone tracks.

5.1.26.1.2 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 650) section for more information.

5.1.26.1.3 METplus Components

This use case utilizes the METplus TCMPRPlotter wrapper to invoke the the MET script tcmpr_plotter.R.

5.1.26.1.4 METplus Workflow

tcmpr_plotter.R is the only tool (script) called in this example.

5.1.26.1.5 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

```
#
#  CONFIGURATION
#
[config]

# Loop over each process in the process list (set in PROCESS_LIST) for all times in the time␣
↪window of
# interest.
LOOP_ORDER = processes

PROCESS_LIST = TCMPRPlotter

# The init time begin and end times, increment
LOOP_BY = INIT
INIT_TIME_FMT = %Y%m
INIT_BEG = 201503
INIT_END = 201503

# This is the step-size. Increment in seconds from the begin time to the end
# time
INIT_INCREMENT = 6H

TCMPR_PLOTTER_TCMPR_DATA_DIR = {INPUT_BASE}/met_test/tc_pairs/{date?fmt=%Y%m}
TCMPR_PLOTTER_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/tcmpr_plots

#TCMPR_PLOTTER_READ_ALL_FILES = True

#  Plot_TCMPR options, if left unset, default values that are
#  pre-defined in the R utility (packaged with MET) will be used.

# Config file used to customize the plot, the tcmpr_customize.conf
```

(continues on next page)

(continued from previous page)

```
# file is used to resize the plot that is produced so that it doesn't
# fill the entire screen.
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 =
# 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 =
TCMPR_PLOTTER_LEAD =
# Default plot is boxplot, unless otherwise indicated.  If box plot is needed
# in addition to other plots, this needs to be indicated.
TCMPR_PLOTTER_PLOT_TYPES = MEAN, MEDIAN
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
```

5.1.26.1.6 MET Configuration

A MET configuration is not needed to run this single wrapper use case.

5.1.26.1.7 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.

5.1.26.1.8 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

5.1.26.1.9 Keywords

Note:

- TCMPRPlotterUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.27 TCPairs

5.1.27.1 TCPairs: Basic Use Case for Extra Tropical Cyclones

met_tool_wrapper/TCPairs/TCPairs_extra_tropical.conf

5.1.27.1.1 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.

5.1.27.1.2 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 658) section for more information.

Data Source: GFS

5.1.27.1.3 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.

5.1.27.1.4 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

5.1.27.1.5 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]

PROCESS_LIST = TCPairs

###
# Time Info
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2014121318
INIT_END = 2014121318
INIT_INCREMENT = 21600 ;; set to every 6 hours=21600 seconds

TC_PAIRS_RUN_ONCE = True

```

(continues on next page)

(continued from previous page)

```
###
# File I/O
###

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_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
###

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

#TC_PAIRS_VALID_INCLUDE =
#TC_PAIRS_VALID_EXCLUDE =
```

(continues on next page)

(continued from previous page)

```
#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_CHECK_DUP =

#TC_PAIRS_INTERP12 =
```

5.1.27.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *TCPairs MET Configuration* (page 243) 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 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}
// init_inc =
```

(continues on next page)

(continued from previous page)

```
${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 = TRUE;

//
// 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
```

(continues on next page)

(continued from previous page)

```
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}

//
// 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}
```

5.1.27.1.7 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.

5.1.27.1.8 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

5.1.27.1.9 Keywords

Note:

- TCPairsToolUseCase
- SBUOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.27.2 TCPairs: Basic Use Case for Tropical Cyclones

met_tool_wrapper/TCPairs/TCPairs_tropical.conf

5.1.27.2.1 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.

5.1.27.2.2 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 666) section for more information.

Data Source: HWRF

5.1.27.2.3 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.

5.1.27.2.4 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

5.1.27.2.5 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]

PROCESS_LIST = TCPairs

###
# Time Info
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2018083006
INIT_END = 2018083018
INIT_INCREMENT = 21600

#TC_PAIRS_SKIP_LEAD_SEQ = False
```
(continues on next page)

(continued from previous page)

```
LOOP_ORDER = times

TC_PAIRS_RUN_ONCE = False


###
# File I/O
###

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_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
###

TC_PAIRS_CONFIG_FILE = {PARM_BASE}/met_config/TCPairsConfig_wrapped

TC_PAIRS_INIT_INCLUDE =
TC_PAIRS_INIT_EXCLUDE =

TC_PAIRS_INIT_BEG =
TC_PAIRS_INIT_END =

#TC_PAIRS_VALID_INCLUDE =
#TC_PAIRS_VALID_EXCLUDE =

#TC_PAIRS_WRITE_VALID =
```

(continues on next page)

(continued from previous page)

```
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_CHECK_DUP =

#TC_PAIRS_INTERP12 =
```

5.1.27.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *TCPairs MET Configuration* (page 243) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// 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}
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
```

(continues on next page)

(continued from previous page)

```
${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.
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
```

(continues on next page)

(continued from previous page)

```
//
//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 = TRUE;

//
// 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;
```

(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 = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.27.2.7 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.

5.1.27.2.8 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

5.1.27.2.9 Keywords

Note:

- TCPairsToolUseCase
- DTCOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.28 TCRMW

5.1.28.1 TCRMW: Basic Use Case

met_tool_wrapper/TCRMW/TCRMW.conf

5.1.28.1.1 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.

5.1.28.1.2 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 672) section for more information.

5.1.28.1.3 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.

5.1.28.1.4 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

5.1.28.1.5 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

```
#
#  CONFIGURATION
#
[config]

# Looping by times: steps through each 'task' in the PROCESS_LIST for each
# defined time, and repeats until all times have been evaluated.
LOOP_ORDER = times
```

(continues on next page)

(continued from previous page)

```
# 'Tasks' to be run
PROCESS_LIST = TCRMW

LOOP_BY = INIT

# The init time begin and end times, increment, and last init hour.
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2016092900
INIT_END = 2016092900

# This is the step-size. Increment in seconds from the begin time to the end time
# set to 6 hours = 21600 seconds
INIT_INCREMENT = 21600

LOG_TC_RMW_VERBOSITY = 2

TC_RMW_CONFIG_FILE = {CONFIG_DIR}/TCRMWConfig_wrapped

MODEL = fv3

#TC_RMW_DESC =

BOTH_VAR1_NAME = PRMSL
BOTH_VAR1_LEVELS = L0

BOTH_VAR2_NAME = TMP
BOTH_VAR2_LEVELS = P1000, P900, P800, P700, P500, P100

# The following variables set values in the MET
# configuration file used by this example
# Leaving these values commented will use the value
# found in the default MET configuration file
#TC_RMW_REGRID_METHOD = NEAREST
#TC_RMW_REGRID_WIDTH = 1
#TC_RMW_REGRID_VLD_THRESH = 0.5
#TC_RMW_REGRID_SHAPE = SQUARE

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
```

(continues on next page)

(continued from previous page)

```
#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 =

#
#  DIRECTORIES
#
[dir]

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/met_config

# Location of input track data directory for DECK data
#TC_RMW_DECK_INPUT_DIR = /d1/projects/MET/MET_test_data/unit_test/tc_data/adeck
TC_RMW_DECK_INPUT_DIR = {INPUT_BASE}/met_test/new/tc_data/adeck

#TC_RMW_INPUT_DIR = /d1/projects/MET/MET_test_data/unit_test/model_data/grib2/gfs_fv3
TC_RMW_INPUT_DIR = {INPUT_BASE}/met_test/new/model_data/grib2/gfs_fv3

TC_RMW_OUTPUT_DIR = {OUTPUT_BASE}/met_tool_wrapper/TCRMW

[filename_templates]
#TC_RMW_DECK_TEMPLATE = a{basin?fmt=%s}{cyclone?fmt=%s}{date?fmt=%Y}.dat
TC_RMW_DECK_TEMPLATE = aal14{date?fmt=%Y}_short.dat

#TC_RMW_INPUT_TEMPLATE = gfs.t00z.pgrb2.0p25.f144
TC_RMW_INPUT_TEMPLATE = gfs.subset.t00z.pgrb2.0p25.f*

TC_RMW_OUTPUT_TEMPLATE = tc_rmw_aal14{date?fmt=%Y}.nc
```

5.1.28.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *TCRMW MET Configuration* (page 250) 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}
```

5.1.28.1.7 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.

5.1.28.1.8 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

5.1.28.1.9 Keywords

Note:

- TCRMWToolUseCase
- GRIB2FileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.29 TCStat

5.1.29.1 TCStat: Basic Use Case

met_tool_wrapper/TCStat/TCStat.conf

5.1.29.1.1 Scientific Objective

Summarize and stratify the data from TC-Pairs track and intensity data

5.1.29.1.2 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 682) section for more information.

Data Source: Unknown

5.1.29.1.3 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.

5.1.29.1.4 METplus Workflow

TCStat is the only tool called in this example. It processes the following TCST run times:

TCST: 2015030100

5.1.29.1.5 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

```
#
#  CONFIGURATION
#
[config]
# set looping method to processes-each 'task' in the process list runs to
# completion (for all init times) before the next 'task' is run
LOOP_ORDER = processes

# List of 'tasks' to run
PROCESS_LIST = TCStat

LOOP_BY = INIT

# The init time begin and end times, increment, and last init hour.
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2019103112
INIT_END = 2019103112

# This is the step-size. Increment in seconds from the begin time to the end time
# set to 6 hours = 21600 seconds
INIT_INCREMENT = 6H

#
#  DIRECTORIES
#
# TC-Stat input data (-lookin argument)
# uses output from tc-pairs
TC_STAT_LOOKIN_DIR = {INPUT_BASE}/met_test/tc_pairs

# TC-Stat output data (creates .tcst ASCII files which can be read or used as
# input to TCMPR_Plotter_wrapper (the Python wrapper to plot_tcmpr.R) to create plots.
TC_STAT_OUTPUT_DIR = {OUTPUT_BASE}/tc_stat

TC_STAT_OUTPUT_TEMPLATE = job.out

# Leave blank or remove to use wrapped config file in parm/met_config
TC_STAT_CONFIG_FILE = {PARM_BASE}/met_config/TCStatConfig_wrapped


#              !!!!!!!IMPORTANT!!!!!!
# Please refer to the README_TC located in ${MET_INSTALL_DIR}/share/met/config
```

(continues on next page)

(continued from previous page)

```
# for details on setting up your analysis jobs.

# For arithmetic expressions such as:
# -column 'ABS(AMSLP-BMSLP)', enclose the expression in ''. Notice that there are no
# whitespaces within the arithmetic expression.  White spaces are to be used to
# separate options from values (e.g. -job summary -by AMODEL,LEAD,AMSLP -init_hour 00 -
↪column 'AMSLP-BMSLP').
# eg. -lookin {OUTPUT_BASE}/tc_pairs -job filter -dump_row {OUTPUT_BASE}/tc_stat_filter.out -
↪basin ML -init_hr 00
# or -lookin {OUTPUT_BASE}/tc_pairs -job summary -by AMODEL,LEAD -column AMSLP -column AMAX_
↪WIND -column 'ABS(AMAX_WIND-BMAX_WIND)' -out {OUTPUT_BASE}/tc_stat/tc_stat_summary.tcst

# Define the job filter via TC_STAT_JOB_ARGS.
# Separate each option and value with whitespace, and each job with a whitespace.
# No whitespace within arithmetic expressions or lists of items
# (e.g. -by AMSLP,AMODEL,LEAD -column '(AMAX_WIND-BMAX_WIND)')
# Enclose your arithmetic expressions with '' and separate each job
# by whitespace:
#  -job filter -dump_row /path/to,  -job summary -line_type TCMPR  -column 'ABS(AMAX_WIND-
↪BMAX_WIND)' -out {OUTPUT_BASE}/tc_stat/file.tcst

TC_STAT_JOB_ARGS = -job summary -line_type TCMPR -column 'ASPEED' -dump_row {TC_STAT_OUTPUT_
↪DIR}/tc_stat_summary.tcst


#
#  FILL in the following values if running multiple jobs which
#  requires a MET tc_stat config file.
#
# These all map to the options in the default TC-Stat config file, except these
# are pre-pended with TC_STAT to avoid clashing with any other similarly
# named options from other MET tools (eg TC_STAT_AMODEL corresponds to the
# amodel option in the default MET tc-stat config file, whereas AMODEL
# corresponds to the amodel option in the MET tc-pairs config file).

# Stratify by these columns:
TC_STAT_AMODEL =
TC_STAT_BMODEL =
TC_STAT_DESC =
TC_STAT_STORM_ID =
TC_STAT_BASIN =
TC_STAT_CYCLONE =
TC_STAT_STORM_NAME =

# Stratify by init times via a comma-separate list of init times to
```

(continues on next page)

(continued from previous page)

```
# include or exclude.  Time format defined as YYYYMMDD_HH or YYYYMMDD_HHmmss
TC_STAT_INIT_BEG = 20150301
TC_STAT_INIT_END = 20150304
TC_STAT_INIT_INCLUDE =
TC_STAT_INIT_EXCLUDE =
TC_STAT_INIT_HOUR = 00

# Stratify by valid times via a comma-separate list of valid times to
# include or exclude.  Time format defined as YYYYMMDD_HH or YYYYMMDD_HHmmss
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 =

# Stratify by the valid time and lead time via comma-separated list of
# times in format HH[MMSS]
TC_STAT_VALID_HOUR =
TC_STAT_LEAD =

# Stratify over the watch_warn column in the tcst file.  Setting this to
# 'ALL' will match HUWARN, HUWATCH, TSWARN, TSWATCH
TC_STAT_TRACK_WATCH_WARN =

# Stratify by applying thresholds to numeric data columns.  Specify with
# comma-separated list of column names and thresholds to be applied.
# The length of TC_STAT_COLUMN_THRESH_NAME should be the same as
# TC_STAT_COLUMN_THRESH_VAL.
TC_STAT_COLUMN_THRESH_NAME =
TC_STAT_COLUMN_THRESH_VAL =

# Stratify by a list of comma-separated columns names and values corresponding
# to non-numeric data columns of the values of interest.
TC_STAT_COLUMN_STR_NAME =
TC_STAT_COLUMN_STR_VAL =

# Stratify by applying thresholds to numeric data columns only when lead=0.
# If lead=0 and the value does not meet the threshold, discard the entire
# track.  The length of TC_STAT_INIT_THRESH_NAME must equal the length of
# TC_STAT_INIT_THRESH_VAL.
TC_STAT_INIT_THRESH_NAME =
TC_STAT_INIT_THRESH_VAL =
```

(continues on next page)

(continued from previous page)

```
# Stratify by applying thresholds to numeric data columns only when lead = 0.
# If lead = 0 but the value doesn't meet the threshold, discard the entire
# track.
TC_STAT_INIT_STR_NAME =
TC_STAT_INIT_STR_VAL =

# Excludes any points where distance to land is <=0. When 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.
TC_STAT_WATER_ONLY =

# TRUE or FALSE.  To specify whether only those track points occurring near
# landfall should be retained. Landfall is the last bmodel track point before
# the distance to land switches from water to land.
TC_STAT_LANDFALL =

# Define the landfall retention window, which is defined as the hours offset
# from the time of landfall. Format is in HH[MMSS]. Default TC_STAT_LANDFALL_BEG
# is set to -24, and TC_STAT_LANDFALL_END is set to 00
TC_STAT_LANDFALL_BEG = -24
TC_STAT_LANDFALL_END = 00

# Specify whether only those track points common to both the ADECK and BDECK
# tracks should be written out
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 =

# IMPORTANT  Refer to the README_TC for details on setting up analysis
# jobs (located in {MET_INSTALL_DIR}/share/met/config
```

5.1.29.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *TCStat MET Configuration* (page 256) 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.
// May add using the "-basin" job command option.
//
${METPLUS_BASIN}

//
// Stratify by the CYCLONE column.
// May add using the "-cyclone" job command option.
```

(continues on next page)

(continued from previous page)

```
//
${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_INCLUDE}
${METPLUS_INIT_EXCLUDE}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INCLUDE}
${METPLUS_VALID_EXCLUDE}

//
// 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}
```

(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 =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_INIT_STR_EXC_NAME}

//init_str_exc_val =
${METPLUS_INIT_STR_EXC_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}

//
// Array of TCStat analysis jobs to be performed on the filtered data
//
${METPLUS_JOBS}

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.1.29.1.7 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.

5.1.29.1.8 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

5.1.29.1.9 Keywords

Note:

- TCStatToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.1.30 UserScript

5.1.30.1 UserScript: Run Once Per Init Use Case

met_tool_wrapper/UserScript/UserScript_run_once_per_init.conf

5.1.30.1.1 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.

5.1.30.1.2 Datasets

Input: Empty test files from the METplus repository

5.1.30.1.3 METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command that is specified by the user.

5.1.30.1.4 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

5.1.30.1.5 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

```
# UserScript wrapper example

[config]

```

(continues on next page)

(continued from previous page)

```
# List of applications to run - only UserScript for this case
PROCESS_LIST = UserScript


# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INIT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H%M%S

# Start time for METplus run - must match VALID_TIME_FMT
INIT_BEG = 20141031093015

# End time for METplus run - must match VALID_TIME_FMT
INIT_END = 20141101093015

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0H, 12H, 24H, 120H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# list of strings to loop over for each run time.
# value for each item can be referenced in filename templates with {custom?fmt=%s}
USER_SCRIPT_CUSTOM_LOOP_LIST = nc
```

(continues on next page)

(continued from previous page)

```
USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

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

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/met_tool_wrapper/UserScript/print_file_list.py
```

5.1.30.1.6 MET Configuration

None. UserScript does not use configuration files.

5.1.30.1.7 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.

5.1.30.1.8 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

5.1.30.1.9 Keywords

Note:

- UserScriptUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.30.2 UserScript: Run Once Per Lead Use Case

met_tool_wrapper/UserScript/UserScript_run_once_per_lead.conf

5.1.30.2.1 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.

5.1.30.2.2 Datasets

Input: Empty test files from the METplus repository

5.1.30.2.3 METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command that is specified by the user.

5.1.30.2.4 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

5.1.30.2.5 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

```
# UserScript wrapper example

[config]

# List of applications to run - only UserScript for this case
PROCESS_LIST = UserScript


# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INIT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H%M%S

# Start time for METplus run - must match VALID_TIME_FMT
INIT_BEG = 20141031093015
```

(continues on next page)

(continued from previous page)

```
# End time for METplus run - must match VALID_TIME_FMT
INIT_END = 20141101093015

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0H, 12H, 24H, 120H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# list of strings to loop over for each run time.
# value for each item can be referenced in filename templates with {custom?fmt=%s}
USER_SCRIPT_CUSTOM_LOOP_LIST = nc

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

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

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/met_tool_wrapper/UserScript/print_file_list.py
```

5.1.30.2.6 MET Configuration

None. UserScript does not use configuration files.

5.1.30.2.7 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.

5.1.30.2.8 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

5.1.30.2.9 Keywords

Note:

- UserScriptUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.30.3 UserScript: Run Once For Each Runtime Use Case

met_tool_wrapper/UserScript/UserScript_run_once_for_each.conf

5.1.30.3.1 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.

5.1.30.3.2 Datasets

Input: Empty test files from the METplus repository

5.1.30.3.3 METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command that is specified by the user.

5.1.30.3.4 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

5.1.30.3.5 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

```
# UserScript wrapper example

[config]

# List of applications to run - only UserScript for this case
PROCESS_LIST = UserScript


# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H%M%S

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 20141031093015

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 20141101093015

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0H, 12H, 24H, 120H
```

(continues on next page)

(continued from previous page)

```
# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# list of strings to loop over for each run time.
# value for each item can be referenced in filename templates with {custom?fmt=%s}
USER_SCRIPT_CUSTOM_LOOP_LIST = nc

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_FOR_EACH

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

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/met_tool_wrapper/UserScript/print_file_list.py
```

5.1.30.3.6 MET Configuration

None. UserScript does not use configuration files.

5.1.30.3.7 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.

5.1.30.3.8 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

5.1.30.3.9 Keywords

Note:

- UserScriptUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.30.4 UserScript: Run Once Per Valid Use Case

met_tool_wrapper/UserScript/UserScript_run_once_per_valid.conf

5.1.30.4.1 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.

5.1.30.4.2 Datasets

Input: Empty test files from the METplus repository

5.1.30.4.3 METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command that is specified by the user.

5.1.30.4.4 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

5.1.30.4.5 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

```
# UserScript wrapper example

[config]

# List of applications to run - only UserScript for this case
PROCESS_LIST = UserScript


# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H%M%S

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 20141031093015

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 20141101093015

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0H, 12H, 24H, 120H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
```

(continues on next page)

(continued from previous page)

```
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# list of strings to loop over for each run time.
# value for each item can be referenced in filename templates with {custom?fmt=%s}
USER_SCRIPT_CUSTOM_LOOP_LIST = nc

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

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

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/met_tool_wrapper/UserScript/print_file_list.py
```

5.1.30.4.6 MET Configuration

None. UserScript does not use configuration files.

5.1.30.4.7 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.

5.1.30.4.8 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

5.1.30.4.9 Keywords

Note:

- UserScriptUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.1.30.5 UserScript: Run Once Use Case

met_tool_wrapper/UserScript/UserScript_run_once.conf

5.1.30.5.1 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.

5.1.30.5.2 Datasets

Input: Empty test files from the METplus repository

5.1.30.5.3 METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command that is specified by the user.

5.1.30.5.4 METplus Workflow

UserScript is the only tool called in this example. It runs once with no time information specified.

5.1.30.5.5 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

```
# UserScript wrapper example

[config]

# List of applications to run - only UserScript for this case
PROCESS_LIST = UserScript


# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H%M%S

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 20141031093015
```

(continues on next page)

(continued from previous page)

```
# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 20141101093015

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0H, 12H, 15H, 24H, 120H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# list of strings to loop over for each run time.
# value for each item can be referenced in filename templates with {custom?fmt=%s}
USER_SCRIPT_CUSTOM_LOOP_LIST = nc

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE

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

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/met_tool_wrapper/UserScript/print_file_list.py
```

5.1.30.5.6 MET Configuration

None. UserScript does not use configuration files.

5.1.30.5.7 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.

5.1.30.5.8 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

5.1.30.5.9 Keywords

Note:

- UserScriptUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

5.2 Model Applications

5.2.1 Air Quality and Composition

Data related to all areas of atmospheric composition, including ozone, smoke, dust, AOD and PM2.5

5.2.1.1 EnsembleStat: Using Python Embedding for Aerosol Optical Depth

model_applications/air_quality_and_comp/EnsembleStat_fcstICAP_obsMODIS_aod.conf

5.2.1.1.1 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.

5.2.1.1.2 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 719) section for more information.

5.2.1.1.3 METplus Components

This use case utilizes the METplus EnsembleStat wrapper to read in files using Python Embedding

5.2.1.1.4 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.

5.2.1.1.5 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

```
# Ensemble Stat using Python Embedding Input

[config]

## Configuration-related settings such as the process list, begin and end times, etc.
PROCESS_LIST = EnsembleStat

# Looping by times: steps through each 'task' in the PROCESS_LIST for each
# defined time, and repeats until all times have been evaluated.
LOOP_ORDER = times

# LOOP_BY: Set to INIT to loop over initialization times
LOOP_BY = INIT

# Format of INIT_BEG and INT_END
INIT_TIME_FMT = %Y%m%d%H%M

# Start time for METplus run
INIT_BEG=201608150000

# End time for METplus run
INIT_END=201608150000

# Increment between METplus runs in seconds. Must be >= 60
INIT_INCREMENT=06H

# List of forecast leads to process
LEAD_SEQ = 12H

# Used in the MET config file for:  model, output_prefix
MODEL = ICAP

# Name to identify observation data in output
OBTYPE = NRL_AOD

# The MET ensemble_stat logging level
# 0 quiet to 5 loud, Verbosity setting for MET ensemble_stat output, 2 is default.
# This takes precendence over the general LOG_MET_VERBOSITY set in metplus_logging.conf
#LOG_ENSEMBLE_STAT_VERBOSITY = 2

OBS_ENSEMBLE_STAT_WINDOW_BEGIN = -5400
```

(continues on next page)

(continued from previous page)

```
OBS_ENSEMBLE_STAT_WINDOW_END = 5400

OBS_FILE_WINDOW_BEGIN = 0
OBS_FILE_WINDOW_END = 0

# number of expected members for ensemble. Should correspond with the
# number of items in the list for FCST_ENSEMBLE_STAT_INPUT_TEMPLATE
ENSEMBLE_STAT_N_MEMBERS = 7

# ens.ens_thresh value in the MET config file
# threshold for ratio of valid files to expected files to allow app to run
ENSEMBLE_STAT_ENS_THRESH = 0.1

# Used in the MET config file for: regrid to_grid field
ENSEMBLE_STAT_REGRID_TO_GRID = NONE

ENSEMBLE_STAT_OUTPUT_PREFIX =

ENSEMBLE_STAT_CONFIG_FILE = {PARM_BASE}/met_config/EnsembleStatConfig_wrapped

# ENSEMBLE_STAT_MET_OBS_ERR_TABLE is not required.
# If the variable is not defined, or the value is not set
# than the MET default is used.
#ENSEMBLE_STAT_MET_OBS_ERR_TABLE =

# Ensemble Variables and levels as specified in the ens field dictionary
# of the MET configuration file. Specify as ENS_VARn_NAME, ENS_VARn_LEVELS,
# (optional) ENS_VARn_OPTION
ENS_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

# Forecast Variables and levels as specified in the fcst field dictionary
# of the MET configuration file. Specify as FCST_VARn_NAME, FCST_VARn_LEVELS,
# (optional) FCST_VARn_OPTION
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

# Observation Variables and levels as specified in the obs field dictionary
# of the MET configuration file. Specify as OBS_VARn_NAME, OBS_VARn_LEVELS,
# (optional) OBS_VARn_OPTION
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

ENS_ENSEMBLE_STAT_INPUT_DATATYPE = PYTHON_NUMPY
```

(continues on next page)

(continued from previous page)

```
FCST_ENSEMBLE_STAT_INPUT_DATATYPE = PYTHON_NUMPY

OBS_ENSEMBLE_STAT_INPUT_GRID_DATATYPE = PYTHON_NUMPY

[dir]
# Use case configuration file directory
CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/air_quality_and_comp/EnsembleStat_
↪fcstICAP_obsMODIS_aod

# Forecast model input directory for ensemble_stat
FCST_ENSEMBLE_STAT_INPUT_DIR =

# Point observation input dir for ensemble_stat
OBS_ENSEMBLE_STAT_POINT_INPUT_DIR =

# Grid observation input dir for ensemble_stat
OBS_ENSEMBLE_STAT_GRID_INPUT_DIR = {INPUT_BASE}/model_applications/air_quality_and_comp/aod

# directory containing climatology mean input to EnsembleStat
# Not used in this example
ENSEMBLE_STAT_CLIMO_MEAN_INPUT_DIR =

# directory containing climatology mean input to EnsembleStat
# Not used in this example
ENSEMBLE_STAT_CLIMO_STDEV_INPUT_DIR =

# output directory for ensemble_stat
ENSEMBLE_STAT_OUTPUT_DIR = {OUTPUT_BASE}


[filename_templates]

# FCST_ENSEMBLE_STAT_INPUT_TEMPLATE  - comma separated list of ensemble members
# or a single line, - wildcard characters may be used.

# FCST_ENSEMBLE_STAT_INPUT_TEMPLATE = ????????gep?/d01_{init?fmt=%Y%m%d%H}_02400.grib
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE = 0, 1, 2, 3, 4, 5, 6

OBS_ENSEMBLE_STAT_POINT_INPUT_TEMPLATE =

OBS_ENSEMBLE_STAT_GRID_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to look for climatology input to EnsembleStat relative to ENSEMBLE_STAT_CLIMO_
↪MEAN_INPUT_DIR
# Not used in this example
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

# Template to look for climatology input to EnsembleStat relative to ENSEMBLE_STAT_CLIMO_
↪STDEV_INPUT_DIR
# Not used in this example
ENSEMBLE_STAT_CLIMO_STDEV_INPUT_TEMPLATE =


ENSEMBLE_STAT_OUTPUT_TEMPLATE =
```

5.2.1.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *EnsembleStat MET Configuration* (page 87) 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    = "";

//
// Ensemble product fields to be processed
//
ens = {

   ${METPLUS_ENS_FILE_TYPE}

   ${METPLUS_ENS_THRESH}
   ${METPLUS_ENS_VLD_THRESH}
   ${METPLUS_ENS_OBS_THRESH}

   ${METPLUS_ENS_FIELD}
}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}

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

//
// Neighborhood ensemble probabilities
```

(continues on next page)

(continued from previous page)

```
//
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
${METPLUS_NMEP_SMOOTH_DICT}

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

//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_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)

```
//

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

//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
```

(continues on next page)

(continued from previous page)

```
//
${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}
   sid   = [];
   llpnt = [];
}

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

//
```

(continues on next page)

(continued from previous page)

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

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

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

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

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

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

//
// Ensemble product output types
//
${METPLUS_ENSEMBLE_FLAG_DICT}

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

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

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

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

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

(continues on next page)

(continued from previous page)

```
tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.1.1.7 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.
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)
```

(continues on next page)

(continued from previous page)

```
    #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',
        '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))
```

(continues on next page)

(continued from previous page)

```
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]

    #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',
```

(continues on next page)

(continued from previous page)

```
        '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))
print("Attributes:\t" + repr(attrs))
f.close()
```

5.2.1.1.8 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.

5.2.1.1.9 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

5.2.1.1.10 Keywords

Note:

- EnsembleStatToolUseCase
- PythonEmbeddingFileUseCase
- AirQualityAndCompAppUseCase
- PythonEmbeddingFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.2 Climate

Average long range earth system predictions

5.2.2.1 Grid-Stat: CESM and GFS Analysis CONUS Temp

model_applications/climate/ GridStat_fcstCESM_obsGFS _ConusTemp.conf

5.2.2.1.1 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.

5.2.2.1.2 Datasets

- Forecast dataset: CESM Surface Temperature Data
- Observation dataset: GFS Analysis 2m Temperature

5.2.2.1.3 METplus Components

This use case runs grid_stat to create continuous statistics on tempepratture from the CESM model and observations from the GFS analysis.

5.2.2.1.4 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

5.2.2.1.5 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

```
# GridStat METplus Configuration for the CESM_vs_GFS climate model use case:
[config]

# List of applications to run - only GridStat for this case
PROCESS_LIST = GridStat

# time looping - 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
LOOP_BY = INIT

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG = 2014080100

# End time for METplus run - must match INIT_TIME_FMT
INIT_END = 2014080200

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 86400

# List of forecast leads to process for each run time (init or valid)
LEAD_SEQ = 6, 12

# Order of loops to process data - Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = times
```

(continues on next page)

(continued from previous page)

```
# Location of MET config file to pass to the GridStat
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

# Name to identify model (forecast) data in output
MODEL = CESM

# Name to identify observation data in output (used in output file path)
OBTYPE = GFS_ANALYS

# Name of forecast variable 1, List of levels to evaluate for forecast variable 1, and
# List of thresholds to evaluate for each name/level combination for forecast variable 1
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);

# Name of observation variable, levels, and thresholds
# levels and thresh must be the same length as FCST_VAR1_LEVELS and FCST_VAR1_THRESH
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);

# Set to true to run GridStat separately for each field specified
# Set to false to create one run of GridStat per run time that
#   includes all fields specified.
# Not used for this example
GRID_STAT_ONCE_PER_FIELD = False

GRID_STAT_REGRID_TO_GRID = FCST

GRID_STAT_VERIFICATION_MASK = {FCST_GRID_STAT_INPUT_DIR}/conus_cesm_mask.nc

# Set to true if forecast data is probabilistic
FCST_IS_PROB = false

# Set to true if observation data is probabilistic
#  Only used if configuring forecast data as the 'OBS' input
OBS_IS_PROB = false

# Output prefix set in grid_stat config file
GRID_STAT_OUTPUT_PREFIX={MODEL}_{CURRENT_OBS_NAME}_vs_{OBTYPE}

GRID_STAT_OUTPUT_FLAG_CTC = STAT
```

(continues on next page)

(continued from previous page)

```
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

# End of [config] section and start of [dir] section
[dir]

# directory containing forecast input to GridStat
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/climate/CESM

# directory containing observation input to GridStat
OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/climate/gfs_analysis

# directory containing climatology input to GridStat
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_DIR =

# directory to write output from GridStat
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/climate/CESM_GridStat

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to GridStat relative to FCST_GRID_STAT_INPUT_DIR
FCST_GRID_STAT_INPUT_TEMPLATE = MetPlus.globe.{init?fmt=%Y-%m-%d}-00000.cam.h0.{init?fmt=%Y-
↪%m-%d}-10800.nc

# Template to look for observation input to GridStat relative to OBS_GRID_STAT_INPUT_DIR
OBS_GRID_STAT_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/gfsanl_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H
↪%M}_000.grb2

# Optional subdirectories relative to GRID_STAT_OUTPUT_DIR to write output from GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_MEAN_INPUT_
↪DIR
```

(continues on next page)

(continued from previous page)

```
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =
```

5.2.2.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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}
```

(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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.2.1.7 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.

5.2.2.1.8 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

5.2.2.1.9 Keywords

Note:

- GridStatToolUseCase
- ClimateAppUseCase
- NetCDFFileUseCase
- NCAROrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.2.2 MODE: CESM and GPCP Asian Monsoon Precipitation

model_applications/climate/MODE_fcstCESM_obsGPCP_AsianMonsoonPrecip.conf

5.2.2.2.1 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.

5.2.2.2.2 Datasets

- Forecast dataset: CESM Daily Precipitation
- Observation dataset: GPCP Daily Precipitation

5.2.2.2.3 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.

5.2.2.2.4 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

5.2.2.2.5 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

```
# GridStat METplus Configuration for the CESM_vs_GPCP climate model use case:
[config]
# List of applications to run - only GridStat for this case
PROCESS_LIST = Mode

LOOP_BY = INIT

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG = 2014060100

# End time for METplus run - must match INIT_TIME_FMT
INIT_END = 2014060200

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 86400

# List of forecast leads to process for each run time (init or valid)
LEAD_SEQ = 24, 48

# Order of loops to process data - Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = times


# Forecast Reflectivity Variable Information
MODEL = CESM
FCST_VAR1_NAME = PRECT
```

(continues on next page)

(continued from previous page)

```
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


# MRMS Reflecivitiy Variable Information
OBTYPE = GPCP
OBS_VAR1_NAME = precip
OBS_VAR1_LEVELS = "(0,*,*)"

MODE_CONFIG_FILE = {PARM_BASE}/met_config/MODEConfig_wrapped

MODE_GRID_RES = 1

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

[dir]
# Directory for CESM data
FCST_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/climate/CESM
```

(continues on next page)

(continued from previous page)

```
# Directory of the MRMS obs
OBS_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/climate/GPCP

# Output Data
MODE_OUTPUT_DIR = {OUTPUT_BASE}/climate/CESM_MODE

[filename_templates]
# Forecast Filename Templates:
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_TEMPLATE = gpcp_v01r03_daily_d{valid?fmt=%Y%m%d?shift=-86400}_c20170530.nc
MODE_OUTPUT_TEMPLATE = {init?fmt=%Y_%m_%d_%H%M%S}
MODE_VERIFICATION_MASK_TEMPLATE = {FCST_MODE_INPUT_DIR}/asia_monsoon_cesm_mask.nc
```

5.2.2.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *MODE MET Configuration* (page 159) 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}

//
// Forecast and observation fields to be verified
//
fcst = {
   ${METPLUS_FCST_FIELD}
```

(continues on next page)

(continued from previous page)

```
    ${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}
}

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}
}

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

//
// 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
//
```

(continues on next page)

(continued from previous page)

```
//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 = (
      (  0.00, 1.0 )
      (  0.10, 1.0 )
      (  0.75, 0.0 )
   );

   corner   = 0.8;
```

(continues on next page)

(continued from previous page)

```
   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
//
met_data_dir = "MET_BASE";

fcst_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
   plot_min         = 0.0;
```

(continues on next page)

(continued from previous page)

```
   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}


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

shift_right = 0;    //  grid squares

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

${METPLUS_OUTPUT_PREFIX}
```

(continues on next page)

(continued from previous page)

```
//version        = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.2.2.7 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.

5.2.2.2.8 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

5.2.2.2.9 Keywords

Note:

- MODEToolUseCase
- ClimateAppUseCase
- NetCDFFileUseCase
- NCAROrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.3 Convection Allowing Models

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)

5.2.3.1 MODE: Brightness Temperature Verification

model_applications/ convection_allowing_model/ MODE_fcstFV3_obsGOES_BrightnessTemp.conf

5.2.3.1.1 Scientific Objective

To provide statistical inforation on regions of low brightness temperatures, defined by creating objects, in the FV3 model compared to GOES satellite.

5.2.3.1.2 Datasets

- Forecast dataset: FV3 Model member data
- Observation dataset: GOES Brightness Temperature

5.2.3.1.3 METplus Components

This use case runs MODE to create object statistics on brightness temperatures below 235 K.

5.2.3.1.4 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

5.2.3.1.5 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/convection_allowing_models/MODE_fcstFV3_obsGOES_BrightnessTemp.conf

```
[config]
# Loop by model initialization
LOOP_BY = init

INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run
INIT_BEG = 2019052100

# End time for METplus run
INIT_END = 2019052100

# Increment between METplus runs in seconds. Must be >= 60
INIT_INCREMENT = 3600

LOOP_ORDER = processes

PROCESS_LIST = MODE(lsm1), MODE(mp1)

```

(continues on next page)

(continued from previous page)

```
# Sequence of leads to process
LEAD_SEQ = 1,2

FCST_IS_PROB = false

# MODE variables for the forecast and observations
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_FCST_CENSOR_THRESH = <=0
MODE_FCST_CENSOR_VAL = 9999

MODE_OBS_CENSOR_THRESH = <=0
MODE_OBS_CENSOR_VAL = 9999


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


# Forecast Brightness Temperature Variable Information
MODEL = FV3_core
FCST_VAR1_NAME = SBTA1613_topofatmosphere
```

(continues on next page)

(continued from previous page)

```
FCST_VAR1_LEVELS = "(*,*)"
FCST_VAR1_OPTIONS = file_type = NETCDF_MET;

# Obs GOES Brightness Temperature Variable Information
OBTYPE = GOES
OBS_VAR1_NAME = channel_13_brightness_temperature
OBS_VAR1_LEVELS = "(*,*)"
OBS_VAR1_OPTIONS = file_type = NETCDF_MET;

#CONFIG_DIR={PARM_BASE}/use_cases/model_applications/convection_allowing_models/MODE_fcstFV3_
↪obsGOES_BrightnessTemp
MODE_CONFIG_FILE = {PARM_BASE}/met_config/MODEConfig_wrapped
MODE_REGRID_TO_GRID = NONE

MODE_OUTPUT_PREFIX = FV3_core_{instance}

[dir]
# Directory for FV3 data
FCST_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/brightness_
↪temperature

# Directory of the GOES obs
OBS_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/brightness_
↪temperature

# Output Data Location
MODE_OUTPUT_DIR = {OUTPUT_BASE}/convection_allowing_models/brightness_temperature

[filename_templates]
# Forecast Filename Template
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 Filename Template
OBS_MODE_INPUT_TEMPLATE = {valid?fmt=%Y_%m_%d}_141/remap_GOES-16.{valid?fmt=%Y%m%d}.{valid?
↪fmt=%H%M%S}.nc
```

5.2.3.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *MODE MET Configuration* (page 159) 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 = {
```

(continues on next page)

(continued from previous page)

```
${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}

//
// 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}
}

obs = {
   ${METPLUS_OBS_FIELD}
```

(continues on next page)

(continued from previous page)

```
   ${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}
}

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

//
// 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
//
```

(continues on next page)

(continued from previous page)

```
//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 = (
      (    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;
```

(continues on next page)

(continued from previous page)

```
   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;
}

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
//
```

(continues on next page)

(continued from previous page)

```
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";

tmp_dir = "${MET_TMP_DIR}";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.3.1.7 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/convection_
↪allowing_models/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/convection_
↪allowing_models/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.

5.2.3.1.8 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 convection_allowing_models/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

5.2.3.1.9 Keywords

Note:

- MODEToolUseCase
- MODEToolUseCase
- ConvectionAllowingModelsAppUseCase
- NetCDFFileUseCase
- NOAAEMCOrgUseCase
- NOAAHWTOrgUseCase
- ValidationUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/convection_allowing_models-MODE_fcstFV3_obsGOES_BrightnessTemp.png

Total running time of the script: (0 minutes 0.000 seconds)

5.2.3.2 Grid-Stat: Brightness Temperature Distance Maps

model_applications/ convection_allowing_model/ GridStat_fcstFV3_obsGOES_BrightnessTempDmap.conf

5.2.3.2.1 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.

5.2.3.2.2 Datasets

- Forecast dataset: FV3 Model member data
- Observation dataset: GOES Brightness Temperature

5.2.3.2.3 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.

5.2.3.2.4 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

5.2.3.2.5 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/convection_allowing_models/GridStat_fcstFV3_obsGOES_BrightnessTempDmap.

```
[config]
# Loop by model initialization
LOOP_BY = init

INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run
INIT_BEG = 2019052100

# End time for METplus run
INIT_END = 2019052100

# Increment between METplus runs in seconds. Must be >= 60
INIT_INCREMENT = 3600
```

(continues on next page)

(continued from previous page)

```
LOOP_ORDER = processes

PROCESS_LIST = GridStat(lsm1), GridStat(mp1)

# Sequence of leads to process
LEAD_SEQ = 1,2

FCST_IS_PROB = false

# Grid stat variables
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;

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

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


[dir]
# Input and Output Diretory of the object data
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/
↪brightness_temperature
OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/
↪brightness_temperature

#Grid Stat output data location
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/convection_allowing_models/brightness_temperature/grid_
↪stat
```

(continues on next page)

(continued from previous page)

```
[filename_templates]
# Forecast Filename Template
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 Filename Template
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
```

5.2.3.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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     = "";
}

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

(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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.3.2.7 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/convection_
↪allowing_models/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/convection_
↪allowing_models/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.

5.2.3.2.8 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 convection_allowing_models/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_0000
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

5.2.3.2.9 Keywords

Note:

- GridStatToolUseCase
- ConvectionAllowingModelsAppUseCase
- NetCDFFileUseCase
- NOAAEMCOrgUseCase
- NOAAHWTOrgUseCase
- ValidationUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/convection_allowing_models-GridStat_fcstFV3_obsGOES_BrightnessTempD

Total running time of the script: (0 minutes 0.000 seconds)

5.2.3.3 MODE/Grid-Stat: Brightness Temperature Verification and Distance Maps

model_applications/ convection_allowing_model/ MODE_fcstFV3_obsGOES_BrightnessTempObjs.conf

5.2.3.3.1 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

5.2.3.3.2 Datasets

- Forecast dataset: FV3 Model member data
- Observation dataset: GOES Brightness Temperature

5.2.3.3.3 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.

5.2.3.3.4 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

5.2.3.3.5 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/convection_allowing_models/MODE_fcstFV3_obsGOES_BrightnessTempObjs.con

```
[config]
# Loop by model initialization
LOOP_BY = init

INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run
INIT_BEG = 2019052100

# End time for METplus run
INIT_END = 2019052100

# Increment between METplus runs in seconds. Must be >= 60
INIT_INCREMENT = 3600

LOOP_ORDER = processes

PROCESS_LIST = MODE(lsm1), MODE(mp1), GridStat(lsm1), GridStat(mp1)

# Sequence of leads to process
LEAD_SEQ = 1,2

FCST_IS_PROB = false

# MODE variables for the forecast and observations
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/convection_allowing_models/brightness_
↪temperature/CentUS.nc

# Forecast Brightness Temperature Variable Information
MODEL = FV3_core
FCST_MODE_VAR1_NAME = SBTA1613_topofatmosphere
FCST_MODE_VAR1_LEVELS = "(*,*)"
FCST_MODE_VAR1_OPTIONS = file_type = NETCDF_MET;

# Obs GOES Brightness Temperature Variable Information
OBTYPE = GOES
OBS_MODE_VAR1_NAME = channel_13_brightness_temperature
OBS_MODE_VAR1_LEVELS = "(*,*)"
OBS_MODE_VAR1_OPTIONS = file_type = NETCDF_MET;


MODE_CONFIG_FILE = {PARM_BASE}/met_config/MODEConfig_wrapped

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_FCST_CENSOR_THRESH = <=0
MODE_FCST_CENSOR_VAL = 9999

MODE_OBS_CENSOR_THRESH = <=0
MODE_OBS_CENSOR_VAL = 9999

MODE_MASK_MISSING_FLAG = BOTH

MODE_MASK_POLY_FLAG = BOTH

MODE_WEIGHT_CENTROID_DIST = 4.0
```

(continues on next page)

(continued from previous page)

```
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}


# Grid stat variables
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;

GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

GRID_STAT_OUTPUT_FLAG_NBRCTC = BOTH
GRID_STAT_OUTPUT_FLAG_DMAP = BOTH

GRID_STAT_NC_PAIRS_FLAG_DISTANCE_MAP = TRUE

GRID_STAT_OUTPUT_PREFIX = FV3_core_{instance}

[dir]
# Directory for FV3 data
FCST_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/brightness_
↪temperature
```

(continues on next page)

(continued from previous page)

```
# Directory of the GOES obs
OBS_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/brightness_
↪temperature

# MODE Output Data Location
MODE_OUTPUT_DIR = {OUTPUT_BASE}/convection_allowing_models/brightness_temperature/mode

# Input and Output Diretory of the object data
FCST_GRID_STAT_INPUT_DIR = {MODE_OUTPUT_DIR}
OBS_GRID_STAT_INPUT_DIR = {FCST_GRID_STAT_INPUT_DIR}

#Grid Stat output data location
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/convection_allowing_models/brightness_temperature/grid_
↪stat_obj


[filename_templates]
# Forecast Filename Template
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 Filename Template
OBS_MODE_INPUT_TEMPLATE = {valid?fmt=%Y_%m_%d}_141/remap_GOES-16.{valid?fmt=%Y%m%d}.{valid?
↪fmt=%H%M%S}.nc

# Grid stat forecast filename template
FCST_GRID_STAT_INPUT_TEMPLATE = mode_{MODE_OUTPUT_PREFIX}_{lead?fmt=%HH}0000L_{valid?fmt=%Y%m
↪%d}_{valid?fmt=%H%M%S}V_NAA_obj.nc

# Grid stat obs filename template
OBS_GRID_STAT_INPUT_TEMPLATE = {FCST_GRID_STAT_INPUT_TEMPLATE}
```

5.2.3.3.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *MODE MET Configuration* (page 159) 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}

//
// 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}
}

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}
}
```

(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}
```

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.3.3.7 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/convection_
↪allowing_models/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/convection_
↪allowing_models/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.

5.2.3.3.8 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 convection_allowing_models/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_2019

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_2019
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_00
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

5.2.3.3.9 Keywords

Note:

- MODEToolUseCase
- GridStatToolUseCase
- ConvectionAllowingModelsAppUseCase
- NetCDFFileUseCase
- NOAAEMCOrgUseCase
- NOAAHWTOrgUseCase
- ValidationUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/convection_allowing_models-MODE_fcstFV3_obsGOES_BrightnessTempObj

Total running time of the script: (0 minutes 0.000 seconds)

5.2.3.4 Point2Grid: Calculate Practically Perfect Probabilities

model_applications/ convection_allowing_models/ Point2Grid_obsLSR_ObsOnly_PracticallyPerfect.conf

5.2.3.4.1 Scientific Objective

To use storm reports as observations to calculate Practically Perfect probabilities.

5.2.3.4.2 Datasets

Relevant information about the datasets that would be beneficial include:

- Observation dataset: Local Storm Reports

5.2.3.4.3 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

5.2.3.4.4 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

5.2.3.4.5 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/convection_allowing_models/Point2Grid_obsLSR_ObsOnly_PracticallyPerfect.con

```
[config]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - Ascii2nc and Point2Grid
```

(continues on next page)

(continued from previous page)

```
PROCESS_LIST = ASCII2NC, Point2Grid, RegridDataPlane

# time looping - 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
LOOP_BY = INIT

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG = 2020020500

# End time for METplus run - must match INIT_TIME_FMT
INIT_END = 2020020500

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 24H

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 12H

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Time relative to valid time (in seconds if no units are specified) to allow files to be␣
↪considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
ASCII2NC_FILE_WINDOW_BEGIN = 0
ASCII2NC_FILE_WINDOW_END = 0
```

(continues on next page)

(continued from previous page)

```
# Time relative to each input file's valid time (in seconds if no units are specified) for␣
↪data within the file to be
# considered valid.
ASCII2NC_WINDOW_BEGIN = 0
ASCII2NC_WINDOW_END = 0

# Value to pass with the -format argument to ascii2nc. See MET User's Guide for more␣
↪information
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



# Verbosity of MET output - overrides LOG_VERBOSITY for Point2Grid only
# POINT2GRID_VERBOSITY = 1

# Time relative to valid time (in seconds if no units are specified) to allow files to be␣
↪considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
POINT2GRID_FILE_WINDOW_BEGIN = 0
POINT2GRID_FILE_WINDOW_END = 0

# Value to pass with the -to_grid See MET User's Guide for more information
POINT2GRID_REGRID_TO_GRID = G211

# Value to pass with the -field string. See MET User's Guide for more information
# FIELD and LEVEL both end up in the -field string
POINT2GRID_INPUT_FIELD =Fscale
POINT2GRID_INPUT_LEVEL =

# Value to pass with the -qc argument
POINT2GRID_QC_FLAGS = 0
```

(continues on next page)

(continued from previous page)

```
# Value to pass with the -adp argument - This is a file name with GOES Aerosol Detection␣
↪Product data
POINT2GRID_ADP =

# Value to pass with the -method argumen - Default is UW_MEAN, other examples are
POINT2GRID_REGRID_METHOD = MAX

# Value to pass with the -gaussian-dx argument - Distance interval for gaussian smoothing
# Default is 81.271
POINT2GRID_GAUSSIAN_DX = 81.271

# Value to pass with the -gaussian-radius argument - radius of influence for the gaussian␣
↪smoothing
# Default is 120
POINT2GRID_GAUSSIAN_RADIUS = 120

# Value to pass with the -prob_cat_thresh argument - threshold for probability of occurrence
POINT2GRID_PROB_CAT_THRESH =

# Value to pass with the -vld_thresh argument - threshold for percentage of valid data .5␣
↪default
POINT2GRID_VLD_THRESH =


# Regrid Data Plane
OBS_REGRID_DATA_PLANE_RUN = True

REGRID_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Fscale_mask

# Level of input field to process
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = "(*,*)"

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = PP_probs

# Mask to use for regridding
REGRID_DATA_PLANE_VERIF_GRID = G211

# Method to run regrid_data_plane, not setting this will default to NEAREST
REGRID_DATA_PLANE_METHOD = MAXGAUSS

# Regridding width used in regrid_data_plane, not setting this will default to 1
```

(continues on next page)

(continued from previous page)

```
REGRID_DATA_PLANE_WIDTH = 1

# Set Gaussian dx value to add as command line argument - not added if unset or blank
REGRID_DATA_PLANE_GAUSSIAN_DX = 81.271

# Set Gaussian filter radius value to add as command line argument - not added if unset or␣
↪blank
REGRID_DATA_PLANE_GAUSSIAN_RADIUS = 120

# End of [config] section and start of [dir] section
[dir]

# Input/Output directories can be left empty if the corresponding template contains the full␣
↪path to the files
ASCII2NC_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/practically_
↪perfect
POINT2GRID_INPUT_DIR = {OUTPUT_BASE}/model_applications/convection_allowing_models/
↪practically_perfect
POINT2GRID_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/convection_allowing_models/
↪practically_perfect
OBS_REGRID_DATA_PLANE_INPUT_DIR = {POINT2GRID_OUTPUT_DIR}
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {POINT2GRID_OUTPUT_DIR}

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/convection_allowing_models/Point2Grid_
↪obsLSR_ObsOnly_PracticallyPerfect

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for input to ASCII2NC input storm reports
ASCII2NC_INPUT_TEMPLATE = "{CONFIG_DIR}/read_ascii_storm.py {ASCII2NC_INPUT_DIR}/200205_rpts_
↪filtered.csv"
ASCII2NC_OUTPUT_TEMPLATE = {OUTPUT_BASE}/model_applications/convection_allowing_models/
↪practically_perfect/StormReps.{init?fmt=%Y%m%d%H}.nc

# Templates to use for input to Point2Grid from the output of ASCII2NC and output from␣
↪Point2Grid
POINT2GRID_INPUT_TEMPLATE = {OUTPUT_BASE}/model_applications/convection_allowing_models/
↪practically_perfect/StormReps.{init?fmt=%Y%m%d%H}.nc
POINT2GRID_OUTPUT_TEMPLATE = {OUTPUT_BASE}/model_applications/convection_allowing_models/
↪practically_perfect/StormReps_211.{init?fmt=%Y%m%d%H}.nc

#Regrid data plane templates
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = StormReps_211.{init?fmt=%Y%m%d%H}.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = StormReps_211_Probs.{init?fmt=%Y%m%d}.nc
```

5.2.3.4.6 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";
```

(continues on next page)

(continued from previous page)

```
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

5.2.3.4.7 Python Embedding

This use case uses a Python embedding script to read input data

parm/use_cases/model_applications/convection_allowing_models/Point2Grid_obsLSR_ObsOnly_PracticallyPerfect/re

```
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)

# 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)
```

(continues on next page)

(continued from previous page)

```
    "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 appropriate 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"
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
```

(continues on next page)

(continued from previous page)

```
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)))

######################################################################
```

5.2.3.4.8 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/convection_
↪allowing_models/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/convection_
↪allowing_models/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.

5.2.3.4.9 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/convection_allowing_models/practically_perfect/ (relative to **OUTPUT_BASE**) and will contain the following files:

- StormReps_211_Probs.20200205.nc

5.2.3.4.10 Keywords

Note:

- ASCII2NCToolUseCase
- Point2GridUseCase
- RegridDataPlaneToolUseCase
- RegriddingInToolUseCase
- NetCDFFileUseCase
- PythonEmbeddingFileUseCase
- ConvectionAllowingModelsAppUseCase
- NCAROrgUseCase
- ProbabilityGenerationUseCase
- MaskingFeatureUseCase
- HMTOrgUseCase
- HWTOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/convection_allowing_models-Point2Grid_obsLSR_ObsOnly_PracticallyPerfec

Total running time of the script: (0 minutes 0.000 seconds)

5.2.3.5 Ensemble-Stat: Ensemble Statistics using Obs Uncertainty

model_applications/ convection_allowing_models/ EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField.conf

5.2.3.5.1 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.

5.2.3.5.2 Datasets

Relevant information about the datasets that would be beneficial include:

- Forecast dataset: HRRRE data
- Observation dataset: HRRRE data

5.2.3.5.3 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.

5.2.3.5.4 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

5.2.3.5.5 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/convection_allowing_models/EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiFi

```
[config]

## Configuration-related settings such as the process list, begin and end times, etc.
PROCESS_LIST = PB2NC, EnsembleStat

# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG=2018070912

# End time for METplus run - must match INIT_TIME_FMT
INIT_END=2018070912

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT=3600

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0,1,2

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
```

(continues on next page)

(continued from previous page)

```
LOOP_ORDER = times

# Name to identify model (forecast) data in output
MODEL = HRRRE_ens

OBTYPE = ANALYS

# The MET ensemble_stat logging level
# 0 quiet to 5 loud, Verbosity setting for MET output, 2 is default.
# This takes precendence over the general MET logging level set in metplus_logging.conf
#LOG_ENSEMBLE_STAT_VERBOSITY = 3

# MET Configuration files for pb2nc
PB2NC_CONFIG_FILE = {PARM_BASE}/met_config/PB2NCConfig_wrapped

PB2NC_LEVEL_RANGE_END = 255

PB2NC_QUALITY_MARK_THRESH = 3

# if True, pb2nc will skip processing a file if the output already exists
# used to speed up runs and reduce redundancy
PB2NC_SKIP_IF_OUTPUT_EXISTS = True

# These are appended with PB2NC to differentiate the GRID, POLY, and MESSAGE_TYPE for point_
↪stat.
PB2NC_GRID =
PB2NC_POLY =
PB2NC_STATION_ID =
PB2NC_MESSAGE_TYPE = ADPUPA, ADPSFC, AIRCFT, PROFLR

# Leave empty to process all
PB2NC_OBS_BUFR_VAR_LIST = POB, QOB, TOB, ZOB, UOB, VOB, D_DPT, D_WDIR, D_WIND, D_RH, D_MIXR,␣
↪D_PRMSL

# False for no time summary, True otherwise
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
```

(continues on next page)

(continued from previous page)

```
PB2NC_TIME_SUMMARY_VALID_THRESH = 0.0

PB2NC_OBS_WINDOW_BEGIN = -900
PB2NC_OBS_WINDOW_END = 900

ENSEMBLE_STAT_OBS_WINDOW_BEGIN = -900
ENSEMBLE_STAT_OBS_WINDOW_END = 900

# number of expected members for ensemble. Should correspond with the
#  number of items in the list for FCST_ENSEMBLE_STAT_INPUT_TEMPLATE
ENSEMBLE_STAT_N_MEMBERS = 2

# ens.ens_thresh value in the MET config file
# threshold for ratio of valid files to expected files to allow app to run
ENSEMBLE_STAT_ENS_THRESH = 1.0

# Used in the MET config file for: regrid to_grid field
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_ENSEMBLE_FLAG_LATLON = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN = TRUE
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_WEIGHT = FALSE

ENSEMBLE_STAT_CONFIG_FILE = {PARM_BASE}/met_config/EnsembleStatConfig_wrapped


# MET_OBS_ERR_TABLE is not required.
# If the variable is not defined, or the value is not set
# than the MET default is used.
ENSEMBLE_STAT_MET_OBS_ERR_TABLE = {CONFIG_DIR}/obs_error_table_V8.0.txt

# Variables and levels as specified in the field dictionary of the MET
# configuration file. Specify as FCST_VARn_NAME, FCST_VARn_LEVELS,
# (optional) FCST_VARn_OPTION

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
```

(continues on next page)

(continued from previous page)

```
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}
OBS_VAR1_OPTIONS = ens_ssvar_bin_size = 1.0; ens_phist_bin_size = 0.05; wind_thresh = >2.572;

ENSEMBLE_STAT_OUTPUT_PREFIX = HRRRE_F{lead?fmt=%3H}_ADPSFC

[dir]
# Use case config directory
CONFIG_DIR={PARM_BASE}/use_cases/model_applications/convection_allowing_models/EnsembleStat_
↪fcstHRRRE_obsHRRRE_Sfc_MultiField

# input and output directories for pb2nc
PB2NC_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/hrrr_ensemble_
↪sfc/prepbufr
PB2NC_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/convection_allowing_models/EnsembleStat_
↪fcstHRRRE_obsHRRRE_Sfc_MultiField/rap

# input directory for ensemble_stat
FCST_ENSEMBLE_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/
↪hrrr_ensemble_sfc/fcst

# point observation input dir for ensemble_stat (can also set grid obs)
OBS_ENSEMBLE_STAT_POINT_INPUT_DIR = {PB2NC_OUTPUT_DIR}

# output directory for ensemble_stat
ENSEMBLE_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/convection_allowing_models/
↪EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField/EnsembleStat


[filename_templates]

# input and output templates for pb2nc
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_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/{valid?fmt=%Y%m%d%H}.rap.nc

# input ensemble template - comma separated list of ensemble members
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE =
    {init?fmt=%Y%m%d%H}/postprd_mem0001/wrfprs_conus_mem0001_{lead?fmt=%HH}.grib2,
```

(continues on next page)

(continued from previous page)

```
    {init?fmt=%Y%m%d%H}/postprd_mem0002/wrfprs_conus_mem0002_{lead?fmt=%HH}.grib2

# input template for EnsembleStat can also be defined using a single
# member with wildcard characters to find multiple files
#FCST_ENSEMBLE_STAT_INPUT_TEMPLATE =
#    {init?fmt=%Y%m%d%H}/postprd_mem000?/wrfprs_conus_mem000?_{lead?fmt=%HH}.grib2

OBS_ENSEMBLE_STAT_POINT_INPUT_TEMPLATE = {PB2NC_OUTPUT_TEMPLATE}

ENSEMBLE_STAT_VERIFICATION_MASK_TEMPLATE =
    {INPUT_BASE}/model_applications/convection_allowing_models/mask/EAST.nc,
    {INPUT_BASE}/model_applications/convection_allowing_models/mask/WEST.nc,
    {INPUT_BASE}/model_applications/convection_allowing_models/mask/CONUS.nc,
    {INPUT_BASE}/model_applications/convection_allowing_models/mask/LMV.nc

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

5.2.3.5.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *EnsembleStat MET Configuration* (page 87) 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}
```

(continues on next page)

(continued from previous page)

```
//
// 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    = "";

//
// Ensemble product fields to be processed
//
ens = {

   ${METPLUS_ENS_FILE_TYPE}

   ${METPLUS_ENS_THRESH}
   ${METPLUS_ENS_VLD_THRESH}
   ${METPLUS_ENS_OBS_THRESH}

   ${METPLUS_ENS_FIELD}
}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}
```

(continues on next page)

(continued from previous page)

```
//control_id =
${METPLUS_CONTROL_ID}

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

//
// Neighborhood ensemble probabilities
//
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
${METPLUS_NMEP_SMOOTH_DICT}

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

//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_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 ];

//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}

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

//
// Ensemble product output types
//
${METPLUS_ENSEMBLE_FLAG_DICT}

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

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

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

//grid_weight_flag =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_GRID_WEIGHT_FLAG}

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

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

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.3.5.7 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/convection_
↪allowing_models/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/convection_
↪allowing_models/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.

5.2.3.5.8 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/convection_allowing_models/EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_MultiField/EnsembleStat (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

5.2.3.5.9 Keywords

Note:

- EnsembleStatToolUseCase
- ConvectionAllowingModelsAppUseCase
- PB2NCToolUseCase
- prepBUFRFileUseCase
- GRIB2FileUseCase
- NCAROrgUseCase
- EnsembleAppUseCase
- ProbabilityGenerationUseCase
- NOAAGSLOrgUseCase
- DTCOrgUseCase
- ObsUncertaintyUseCase
- MaskingFeatureUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/convection_allowing_models-EnsembleStat_fcstHRRRE_obsHRRRE_Sfc_Mu

Total running time of the script: (0 minutes 0.000 seconds)

5.2.3.6 METdbLoad: Brightness Temperature

model_applications/ convection_allowing_model/ METdbLoad_fcstFV3_obsGoes_BrightnessTemp.conf

5.2.3.6.1 Scientific Objective

Load MET data into a database using the met_db_load.py script found in dtcenter/METdatadb. Specifically, this use case loads distance map output from grid_stat and mode output into a database.

5.2.3.6.2 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 807) section for more information.

5.2.3.6.3 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.

5.2.3.6.4 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

5.2.3.6.5 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]

# METdbLoad for grid_stat distance maps and MODE
```

(continues on next page)

(continued from previous page)

```
PROCESS_LIST = METDbLoad

LOOP_BY = VALID

VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2019052112
VALID_END = 2019052100
VALID_INCREMENT = 12H

LOOP_ORDER = processes

MET_DB_LOAD_RUNTIME_FREQ = RUN_ONCE

MET_DATA_DB_DIR = {METPLUS_BASE}/../METdatadb

# 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

# Location of input files
MET_DB_LOAD_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/convection_allowing_models/
↪METdbLoad_fcstFV3_obsGoes_BrightnessTemp/grid_stat,{INPUT_BASE}/model_applications/
↪convection_allowing_models/METdbLoad_fcstFV3_obsGoes_BrightnessTemp/grid_stat_obj,{INPUT_
↪BASE}/model_applications/convection_allowing_models/METdbLoad_fcstFV3_obsGoes_
↪BrightnessTemp/mode
```

(continues on next page)

(continued from previous page)

5.2.3.6.6 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 154) 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>
```

5.2.3.6.7 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/convection_allowing_
↪models/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/convection_allowing_
↪models/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
```

5.2.3.6.8 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.

5.2.3.6.9 Keywords

Note:

- METdbLoadUseCase
- ConvectionAllowingModelsAppUseCase
- NOAAEMCOrgUseCase
- NOAAHWTOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/convection_allowing_models-METdbLoad_fcstFV3_obsGoes_BrightnessTemp

Total running time of the script: (0 minutes 0.000 seconds)

5.2.3.7 MODE: Hail Verification

model_applications/ convection_allowing_model/ MODE_fcstHRRR_obsMRMS_Hail_GRIB2.conf

5.2.3.7.1 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.

5.2.3.7.2 Datasets

- Forecast dataset: HRRRv4 data
- Observation dataset: MRMS

5.2.3.7.3 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.

5.2.3.7.4 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

5.2.3.7.5 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/convection_allowing_models/MODE_fcstHRRR_obsMRMS_Hail_GRIB2.conf

```
[config]
LOOP_BY = valid
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 2019052902

# End time for METplus run
VALID_END = 2019052903

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 3600

LOOP_ORDER = processes

PROCESS_LIST = Mode

LEAD_SEQ_MAX = 36
LEAD_SEQ_MIN = 12
FCST_IS_PROB = false

# list of forecast generation to process
INIT_SEQ = 0
```

(continues on next page)

(continued from previous page)

```
MODE_QUILT = True

MODE_CONV_RADIUS = 4

MODE_CONV_THRESH = >=0.5

MODE_MERGE_THRESH = >=0.0

MODE_MERGE_FLAG = NONE

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

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_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

# Forecast Reflectivity Variable Information
MODEL = HRRRv4_HAILCAST
FCST_VAR1_NAME = HAIL
FCST_VAR1_LEVELS = L0
FCST_VAR1_OPTIONS = convert(x) = x / 0.0254

# MRMS Reflecivitiy Variable Information
OBTYPE = MRMS
OBS_VAR1_NAME = MESHMax60min
OBS_VAR1_LEVELS = Z500
OBS_VAR1_OPTIONS = convert(x) = MM_to_IN(x);

#CONFIG_DIR={PARM_BASE}/use_cases/model_applications/convection_allowing_models/MODE_
↪fcstHRRR_obsMRMS_Hail_GRIB2
```

(continues on next page)

(continued from previous page)

```
#MODE_CONFIG_FILE = {CONFIG_DIR}/MODEConfig_hailcast
MODE_CONFIG_FILE = {PARM_BASE}/met_config/MODEConfig_wrapped

MODE_REGRID_TO_GRID = FCST
MODE_REGRID_METHOD = MAX
MODE_REGRID_WIDTH = 2

[dir]

# Directory for HRRR data
FCST_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/hrrr_esrl

# Directory of the MRMS obs
OBS_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/hrrr_esrl

# Output Data
MODE_OUTPUT_DIR = {OUTPUT_BASE}/hailtest

METPLUS_CONF = {MODE_OUTPUT_DIR}/metplus_final.conf

[filename_templates]
# Forecast Filename Templates:
FCST_MODE_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/hrrr_esrl_{init?fmt=%Y%m%d%H}f{lead?fmt=%HHH}.
↪grib2
#OBS_MODE_INPUT_TEMPLATE = {valid?fmt=%Y}/{valid?fmt=%m}/{valid?fmt=%d}/mrms.MESH_Max_60min.
↪{valid?fmt=%Y%m%d}_{valid?fmt=%H%M%S}.grib2
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_VERIFICATION_MASK_TEMPLATE = {FCST_MODE_INPUT_DIR}/{init?fmt=%Y%m%d}_hrefv2_
↪subdomainmask.nc
```

5.2.3.7.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVER-WRITTEN 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 68)

Note: See the *MODE MET Configuration* (page 159) 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}

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

//
// Run all permutations of radius and threshold
```

(continues on next page)

(continued from previous page)

```
//
// quilt =
${METPLUS_QUILT}

//
// MODE Multivar boolean combination logic
//
//multivar_logic =
${METPLUS_MULTIVAR_LOGIC}

//
// 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}
}

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}
}

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

(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}
```

5.2.3.7.7 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/convection_
↪allowing_models/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/convection_
↪allowing_models/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.

5.2.3.7.8 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

5.2.3.7.9 Keywords

Note:

- MODEToolUseCase
- ConvectionAllowingModelsAppUseCase
- GRIB2FileUseCase
- RegriddingInToolUseCase
- NOAAHWTOrgUseCase
- NCAROrgUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/convection_allowing_models-MODE_fcstHRRRE_obsMRMS_Hail_GRIB2.pn

Total running time of the script: (0 minutes 0.000 seconds)

5.2.3.8 Grid-Stat: Surrogate Severe and Practically Perfect Evaluation

model_applications/ convection_allowing_model/ GridStat_fcstHRRR_obsPracPerfect _SurrogateSevere.conf

5.2.3.8.1 Scientific Objective

To evaluate the surrogate severe forecasts at predicting Severe weather using the (12Z - 12Z) practically perfect storm reports.

5.2.3.8.2 Datasets

- Forecast dataset: HRRR Surrogate Severe Data
- Observation dataset: Practically Perfect from Local Storm Reports.

5.2.3.8.3 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.

5.2.3.8.4 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

5.2.3.8.5 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/convection_allowing_models/GridStat_fcstHRRR_obsPracPerfect_SurrogateSeve

```
# HRRR Surrogate Severe verified against Practically Perfect

[config]
# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = VALID
```

(continues on next page)

(continued from previous page)

```
# Format of INIT_BEG and INIT_END
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG=2020020612

# End time for METplus run
VALID_END=2020020612

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT=86400

# forecast leads to process
INIT_SEQ = 0
LEAD_SEQ_MIN = 36
LEAD_SEQ_MAX = 36

# Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = times

# List of applications to run
PROCESS_LIST = GridStat

MODEL = HRRR
OBTYPE = PP

# Forecast Variables
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
```

(continues on next page)

(continued from previous page)

```
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 Variables
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}

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}

FCST_IS_PROB = false

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

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

[dir]

# input and output data directories for each application in PROCESS_LIST
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/
↪surrogate_severe_prac_perfect
```

(continues on next page)

(continued from previous page)

```
OBS_GRID_STAT_INPUT_DIR = {FCST_GRID_STAT_INPUT_DIR}

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/convection_allowing_models/surrogate_
↪severe_prac_perfect/grid_stat

[filename_templates]
# format of filenames
# Surrogate Severe
FCST_GRID_STAT_INPUT_TEMPLATE = surrogate_severe_{init?fmt=%Y%m%d}_{lead?fmt=%HHH}V_regrid.nc

# Practically Perfect
OBS_GRID_STAT_INPUT_TEMPLATE = StormReps_211_Probs.{init?fmt=%Y%m%d}.nc
```

5.2.3.8.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.3.8.7 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/convection_
↪allowing_models/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/convection_
↪allowing_models/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.

5.2.3.8.8 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/convection_allowing_models/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

5.2.3.8.9 Keywords

Note:

- GridStatToolUseCase
- ConvectionAllowingModelsAppUseCase
- NetCDFFileUseCase
- NOAAHWTOrgUseCase
- NCAROrgUseCase
- NOAAHMTOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/convection_allowing_models-SS_PP_prob.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.3.9 Grid-Stat: Surrogate Severe and Practically Perfect Probabilistic Evaluation

model_applications/ convection_allowing_models/ GridStat_fcstHRRR_obsPracPerfect _SurrogateSevereProb.conf

5.2.3.9.1 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.

5.2.3.9.2 Datasets

- Forecast dataset: HRRR Surrogate Severe Data
- Observation dataset: Practically Perfect from Local Storm Reports

5.2.3.9.3 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.

5.2.3.9.4 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

5.2.3.9.5 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/convection_allowing_models/GridStat_fcstHRRR_obsPracPerfect_SurrogateSeve

```
# HRRR Surrogate Severe verified against Practically Perfect

[config]
# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = VALID

# Format of INIT_BEG and INIT_END
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG=2020020612

# End time for METplus run
VALID_END=2020020612

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT=86400

# forecast leads to process
INIT_SEQ = 0
LEAD_SEQ_MIN = 36
LEAD_SEQ_MAX = 36

# Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = times
```

(continues on next page)

(continued from previous page)

```
# List of applications to run
PROCESS_LIST = GridStat

MODEL = HRRR
OBTYPE = PP

# Forecast Variables
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 Variables
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}
```

(continues on next page)

(continued from previous page)

```
OBS_VAR4_THRESH = {OBS_VAR1_THRESH}

OBS_VAR5_NAME = {OBS_VAR1_NAME}
OBS_VAR5_LEVELS = {OBS_VAR1_LEVELS}
OBS_VAR5_THRESH = {OBS_VAR1_THRESH}

FCST_IS_PROB = true

FCST_GRID_STAT_INPUT_DATATYPE = NETCDF

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

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

[dir]

# input and output data directories for each application in PROCESS_LIST
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/convection_allowing_models/
↪surrogate_severe_prac_perfect

OBS_GRID_STAT_INPUT_DIR = {FCST_GRID_STAT_INPUT_DIR}

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/convection_allowing_models/surrogate_
↪severe_prac_perfect/grid_stat/prob

[filename_templates]
# format of filenames
# Surrogate Severe
FCST_GRID_STAT_INPUT_TEMPLATE = surrogate_severe_{init?fmt=%Y%m%d}_{lead?fmt=%HHH}V_regrid.nc

# Practically Perfect
OBS_GRID_STAT_INPUT_TEMPLATE = StormReps_211.{init?fmt=%Y%m%d}.nc
```

5.2.3.9.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVER-WRITTEN 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.3.9.7 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/convection_
↪allowing_models/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/convection_
↪allowing_models/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.

5.2.3.9.8 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/convection_allowing_models/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

5.2.3.9.9 Keywords

Note:

- GridStatToolUseCase
- ConvectionAllowingModelsAppUseCase
- NetCDFFileUseCase
- NOAAHWTOrgUseCase
- NCAROrgUseCase
- NOAAHMTOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/convection_allowing_models-SS_PP_prob.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.3.10 Surrogate Severe Calculation: PCPCombine, EnsembleStat, and RegridDataPlane

model_applications/convection_allowing_model/EnsembleStat_fcstHRRR_fcstOnly_SurrogateSevere.conf

5.2.3.10.1 Scientific Objective

Run PCPCombine, EnsembleStat, and RegridDataPlane tools to create surrogate severe probability forecasts (SSPFs) for a given date. SSPFs are a severe weather forecasting tool and is a techniqu used by the Storm Prediction Center (SPC) as well as others. SSPFs are based on updraft helicity (UH; UH = z0 to zt (*) 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).

5.2.3.10.2 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.

5.2.3.10.3 METplus Components

This use case runs the PCPCombine, EnsembleStat, and RegridDataPlane MET tools.

5.2.3.10.4 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 EnsembleStat, and then, finally, RegridDataPlane.

5.2.3.10.5 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]

PROCESS_LIST = PCPCombine, EnsembleStat, RegridDataPlane

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2020020500
INIT_END=2020020500
INIT_INCREMENT=86400

LEAD_SEQ = 36

LOOP_ORDER = processes


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/convection_allowing_models/
↪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}/convection_allowing_models/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


FCST_ENSEMBLE_STAT_INPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE = {FCST_PCP_COMBINE_OUTPUT_TEMPLATE}

ENSEMBLE_STAT_OUTPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}


FCST_REGRID_DATA_PLANE_RUN = True

FCST_REGRID_DATA_PLANE_INPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
FCST_REGRID_DATA_PLANE_INPUT_TEMPLATE = ensemble_stat_{valid?fmt=%Y%m%d}_120000V_ens.nc

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
```

(continues on next page)

(continued from previous page)

```
MODEL = FCST_ens
OBTYPE = ANALYS


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


ENSEMBLE_STAT_CONFIG_FILE = {PARM_BASE}/met_config/EnsembleStatConfig_wrapped

ENSEMBLE_STAT_N_MEMBERS = 1
ENSEMBLE_STAT_ENS_THRESH = 1.0

ENSEMBLE_STAT_REGRID_TO_GRID = G211
ENSEMBLE_STAT_REGRID_METHOD = MAX
ENSEMBLE_STAT_REGRID_WIDTH = 27
ENSEMBLE_STAT_REGRID_VLD_THRESH = 0.0

ENSEMBLE_STAT_DUPLICATE_FLAG = UNIQUE
ENSEMBLE_STAT_SKIP_CONST = True

ENSEMBLE_STAT_CENSOR_THRESH = ==-9999
ENSEMBLE_STAT_CENSOR_VAL = 0.0

ENSEMBLE_STAT_OBS_ERROR_FLAG = True

ENSEMBLE_STAT_MASK_GRID =

ENSEMBLE_STAT_CI_ALPHA = 0.01

ENSEMBLE_STAT_OUTPUT_FLAG_ECNT = NONE
ENSEMBLE_STAT_OUTPUT_FLAG_RPS = NONE
ENSEMBLE_STAT_OUTPUT_FLAG_RHIST = NONE
ENSEMBLE_STAT_OUTPUT_FLAG_PHIST = NONE
ENSEMBLE_STAT_OUTPUT_FLAG_ORANK = NONE
ENSEMBLE_STAT_OUTPUT_FLAG_SSVAR = NONE
ENSEMBLE_STAT_OUTPUT_FLAG_RELP = NONE
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_ENSEMBLE_FLAG_LATLON = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_WEIGHT = FALSE


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



REGRID_DATA_PLANE_ONCE_PER_FIELD = False

REGRID_DATA_PLANE_VERIF_GRID = G211
```

(continues on next page)

(continued from previous page)

```
REGRID_DATA_PLANE_METHOD = MAXGAUSS

REGRID_DATA_PLANE_WIDTH = 1

REGRID_DATA_PLANE_GAUSSIAN_DX = 81.271
REGRID_DATA_PLANE_GAUSSIAN_RADIUS = 120
```

5.2.3.10.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *EnsembleStat MET Configuration* (page 87) 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}
```

(continues on next page)

(continued from previous page)

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

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

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

//
// May be set separately in each "field" entry
//
${METPLUS_CENSOR_THRESH}
${METPLUS_CENSOR_VAL}
cat_thresh    = [];
nc_var_str    = "";

//
// Ensemble product fields to be processed
//
ens = {

   ${METPLUS_ENS_FILE_TYPE}

   ${METPLUS_ENS_THRESH}
   ${METPLUS_ENS_VLD_THRESH}
   ${METPLUS_ENS_OBS_THRESH}

   ${METPLUS_ENS_FIELD}
}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}

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

//
// Neighborhood ensemble probabilities
//
${METPLUS_NBRHD_PROB_DICT}
```

(continues on next page)

(continued from previous page)

```
//
// NMEP smoothing methods
//
${METPLUS_NMEP_SMOOTH_DICT}

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

//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_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}
```

(continues on next page)

(continued from previous page)

```
sid_exc        = [];
obs_thresh     = [ NA ];

//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}

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

//
// Ensemble product output types
//
${METPLUS_ENSEMBLE_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}
```

5.2.3.10.7 Running METplus

The command to run this use case is:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/convection_allowing_
↪models/EnsembleStat_fcstHRRR_fcstOnly_SurrogateSevere.conf
```

5.2.3.10.8 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
#
# **EnsembleStat output**:
#
# * ensemble_stat_20190519_120000V_ens.nc
#
# **RegridDataPlane output**:
#
# * surrogate_severe_20190518_036V_regrid.nc
#
```

5.2.3.10.9 Keywords

Note:

- PCPCombineUseCase
- EnsembleStatUseCase
- RegridDataPlaneUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/convection_allowing_models-EnsembleStat_fcstHRRR_fcstOnly_SurrogateS

Total running time of the script: (0 minutes 0.000 seconds)

5.2.4 Data Assimilation

Observational data used as part of the initial conditions for numerical weather prediction

5.2.4.1 StatAnalysis: JEDI

model_applications/data_assimilation/StatAnalysis_fcstHAFS_obsPrepBufr_JEDI_IODA_interface.conf

5.2.4.1.1 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.

5.2.4.1.2 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 859) section for more information.

5.2.4.1.3 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.

5.2.4.1.4 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

5.2.4.1.5 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

```
# StatAnalysis METplus Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# List of applications to run - only StatAnalysis for this case
PROCESS_LIST = StatAnalysis

# time looping - 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
LOOP_BY = VALID

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H


# Start time for METplus run - must match INIT_TIME_FMT
```

(continues on next page)

(continued from previous page)

```
VALID_BEG=2005080700

# End time for METplus run - must match INIT_TIME_FMT
VALID_END=2005080700

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for StatAnalysis only
LOG_STAT_ANALYSIS_VERBOSITY = 2

# Models to process
# MODELn is the model name to filter for in
#           stat files [required]
# MODELn_OBTYPE is the observation name
#               to filter for the .stat files
#               [required]
# MODELn_STAT_ANALYSIS_LOOKIN_DIR is the directory to search for
#                                 the .stat files in, wildcards (*)
#                                 are okay to search for multiple
#                                 directories and templates like
#                                 {valid?fmt=%Y%m%d%H%M%S} [required]
# MODELn_REFERENCE_NAME is a reference name for MODELn, defaults to
#                       MODELn, it can be used in the file template names
#                       [optional]
MODEL1 = NA
MODEL1_OBTYPE = NA

# Location of MET config file to pass to StatAnalysis
```

(continues on next page)

(continued from previous page)

```
# References CONFIG_DIR from the [dir] section
#STAT_ANALYSIS_CONFIG_FILE = {CONFIG_DIR}/STATAnalysisConfig_wrapped

# stat_analysis job info
STAT_ANALYSIS_JOB_NAME = filter
# 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 CNT -dump_row [dump_row_file] -line_type MPR

# Optional variables for further filtering
# can be blank, single, or multiple values
# if more than one use comma separated list
#
# (FCST)(OBS)_(VALID)(INIT)_HOUR_LIST: HH format (ex. 00, 06, 12)
# (FCST)(OBS)_LEAD_LIST: HH[H][MMSS] format (ex. 00, 06, 120)
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 =
# how to treat items listed in above _LIST variables
# GROUP_LIST_ITEMS: items listed in a given _LIST variable
#                   will be grouped together
# LOOP_LIST_ITEMS:  items listed in a give _LIST variable
#                   will be looped over
# if not listed METplus will treat the list as a group
GROUP_LIST_ITEMS =
```

(continues on next page)

(continued from previous page)

```
LOOP_LIST_ITEMS = MODEL_LIST


# End of [config] section and start of [dir] section
[dir]

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

# Output data directory
STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/data_assimilation/StatAnalysis_
↪HofX

# location of configuration files used by MET applications
CONFIG_DIR = {PARM_BASE}/met_config


# End of [dir] section and start of [filename_templates] section
[filename_templates]
# Optional settings to create templated directory and file name information
# to save files as stat_analysis output as, this is appended to STAT_ANALYSIS_OUTPUT_DIR
# if no template is provided a default filename set in the code will be used
# Use:
# string templates can be set for all the lists being looped over, just
# use and a lower case version of the list, ex. {fcst_valid_hour?fmt=%H}
# or {fcst_var?fmt=%s}
# For looping over models:
# can set MODELn_STAT_ANALYSIS_[DUMP_ROW/OUT_STAT]_TEMPLATE for individual models
# or STAT_ANALYSIS_[DUMP_ROW/OUT_STAT] with {model?fmt=%s}
MODEL1_STAT_ANALYSIS_DUMP_ROW_TEMPLATE = dump.out

#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
```

5.2.4.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *StatAnalysis MET Configuration* (page 218) 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}
${METPLUS_OBS_LEVEL}
```

(continues on next page)

(continued from previous page)

```
${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}
```

5.2.4.1.7 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
```

(continues on next page)

(continued from previous page)

```
###########################################################################
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:
        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]
```

(continues on next page)

(continued from previous page)

```
        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']]

            # 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")
```

(continues on next page)

(continued from previous page)

```
    sys.exit(1)

########################################################################

```

5.2.4.1.8 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.

5.2.4.1.9 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

5.2.4.1.10 Keywords

Note:

- StatAnalysisToolUseCase
- PythonEmbeddingFileUseCase
- TCandExtraTCAppUseCase
- NOAAEMCOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.5 Marine and Cryosphere

Data related to verification involving marine and cryosphere systems, which includes sea-ice

5.2.5.1 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

5.2.5.1.1 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.

5.2.5.1.2 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 881) section for more information.

Data Source: JPL's PODAAC and NCEP's FTPPRD data servers

5.2.5.1.3 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

5.2.5.1.4 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

5.2.5.1.5 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

5.2.5.1.6 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

```
# GridStat METplus Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# List of applications to run - only GridStat for this case
PROCESS_LIST = GridStat

# time looping - 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
LOOP_BY = VALID

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match INIT_TIME_FMT
VALID_BEG=20210503

# End time for METplus run - must match INIT_TIME_FMT
VALID_END=20210503

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0


# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
```

(continues on next page)

(continued from previous page)

```
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for GridStat only
LOG_GRID_STAT_VERBOSITY = 2

# Location of MET config file to pass to GridStat
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

# grid to remap data. Value is set as the 'to_grid' variable in the 'regrid' dictionary
# See MET User's Guide for more information
GRID_STAT_REGRID_TO_GRID = NONE

#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 = AREA

# Name to identify model (forecast) data in output
MODEL = RTOFS

# Name to identify observation data in output
OBTYPE = SMOS

# set the desc value in the GridStat MET config file
GRID_STAT_DESC = NA

# List of variables to compare in GridStat - FCST_VAR1 variables correspond
#  to OBS_VAR1 variables
# Note [FCST/OBS/BOTH]_GRID_STAT_VAR<n>_NAME can be used instead if different evaluations
# are needed for different tools

# Name of forecast variable 1
```

(continues on next page)

(continued from previous page)

```
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/
↪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

# List of levels to evaluate for forecast variable 1
# A03 = 3 hour accumulation in GRIB file
FCST_VAR1_LEVELS =

# List of thresholds to evaluate for each name/level combination for
#  forecast variable 1
FCST_VAR1_THRESH =

#FCST_GRID_STAT_FILE_TYPE =

# Name of observation variable 1
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


# List of levels to evaluate for observation variable 1
# (*,*) is NetCDF notation - must include quotes around these values!
# must be the same length as FCST_VAR1_LEVELS
OBS_VAR1_LEVELS =

# List of thresholds to evaluate for each name/level combination for
#  observation variable 1
OBS_VAR1_THRESH =

#GRID_STAT_MET_CONFIG_OVERRIDES = cat_thresh = [>=0.15];
#BOTH_VAR1_THRESH = >=0.15

#OBS_GRID_STAT_FILE_TYPE =


# Name of climatology variable 1
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="(*,*)";}
```

(continues on next page)

(continued from previous page)

```
# Time relative to valid time (in seconds) to allow files to be considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
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

# MET GridStat neighborhood values
# See the MET User's Guide GridStat section for more information

# width value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_WIDTH = 1

# shape value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

# cov thresh list passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

# Set to true to run GridStat separately for each field specified
# Set to false to create one run of GridStat per run time that
#   includes all fields specified.
GRID_STAT_ONCE_PER_FIELD = False

# Set to true if forecast data is probabilistic
FCST_IS_PROB = false

# Only used if FCST_IS_PROB is true - sets probabilistic threshold
FCST_GRID_STAT_PROB_THRESH = ==0.1

# Set to true if observation data is probabilistic
#  Only used if configuring forecast data as the 'OBS' input
OBS_IS_PROB = false

# Only used if OBS_IS_PROB is true - sets probabilistic threshold
OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = SSS

#GRID_STAT_CLIMO_MEAN_FILE_NAME =
#GRID_STAT_CLIMO_MEAN_FIELD =
#GRID_STAT_CLIMO_MEAN_REGRID_METHOD =
```

(continues on next page)

(continued from previous page)

```
#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 =
#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_OUTPUT_FLAG_FHO = NONE
#GRID_STAT_OUTPUT_FLAG_CTC = NONE
#GRID_STAT_OUTPUT_FLAG_CTS = NONE
#GRID_STAT_OUTPUT_FLAG_MCTC = NONE
#GRID_STAT_OUTPUT_FLAG_MCTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = BOTH
#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
```

(continues on next page)

(continued from previous page)

```
#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


# End of [config] section and start of [dir] section
[dir]
#use case configuration file directory
CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsSMOS_climWOA_sss
# directory containing forecast input to GridStat
FCST_GRID_STAT_INPUT_DIR =

# directory containing observation input to GridStat
OBS_GRID_STAT_INPUT_DIR =

# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_DIR =

# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_STDEV_INPUT_DIR =

# directory to write output from GridStat
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to GridStat relative to FCST_GRID_STAT_INPUT_DIR
FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to look for observation input to GridStat relative to OBS_GRID_STAT_INPUT_DIR
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY
```

(continues on next page)

(continued from previous page)

```
# Optional subdirectories relative to GRID_STAT_OUTPUT_DIR to write output from GridStat
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_MEAN_INPUT_
↪DIR
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_STDEV_INPUT_
↪DIR
# Not used in this exampls
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

# Used to specify one or more verification mask files for GridStat
# Not used for this example
GRID_STAT_VERIFICATION_MASK_TEMPLATE =
```

5.2.5.1.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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
//
```

(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;
   rep_prop = 1.0;
```

(continues on next page)

(continued from previous page)

```
   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
//
gradient = {
```

(continues on next page)

(continued from previous page)

```
   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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.5.1.8 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
"""

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'
```

(continues on next page)

(continued from previous page)

```
####################################################################
# 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'})


# 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')
```

(continues on next page)

(continued from previous page)

```
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,]

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
```

(continues on next page)

(continued from previous page)

```
######################################################################
# 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')
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
```

(continues on next page)

(continued from previous page)

```
    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
    """

    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
```

(continues on next page)

(continued from previous page)

```
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[:,:]
    #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,
```

(continues on next page)

(continued from previous page)

```
                '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")
    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,
                }
```

(continues on next page)

(continued from previous page)

```
            }
    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]
    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
```

5.2.5.1.9 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.

5.2.5.1.10 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

5.2.5.1.11 Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.5.2 GridStat: Python Embedding to read and process ice cover

model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsOSTIA_iceCover.conf

5.2.5.2.1 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.

5.2.5.2.2 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 899) section for more information.

Data Source: JPL's PODAAC and NCEP's FTPPRD data servers

5.2.5.2.3 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

5.2.5.2.4 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

5.2.5.2.5 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

5.2.5.2.6 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

```
# GridStat METplus Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]
```

(continues on next page)

(continued from previous page)

```
# List of applications to run - only GridStat for this case
PROCESS_LIST = GridStat

# time looping - 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
LOOP_BY = VALID

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match INIT_TIME_FMT
VALID_BEG=20210305

# End time for METplus run - must match INIT_TIME_FMT
VALID_END=20210305

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0

GRID_STAT_CUSTOM_LOOP_LIST = north, south

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for GridStat only
```

(continues on next page)

(continued from previous page)

```
#LOG_GRID_STAT_VERBOSITY = 2

# Location of MET config file to pass to GridStat
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

# grid to remap data. Value is set as the 'to_grid' variable in the 'regrid' dictionary
# See MET User's Guide for more information
GRID_STAT_REGRID_TO_GRID = NONE

#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 = AREA

# Name to identify model (forecast) data in output
MODEL = RTOFS

# Name to identify observation data in output
OBTYPE = UKMO

# set the desc value in the GridStat MET config file
GRID_STAT_DESC = NA

# List of variables to compare in GridStat - FCST_VAR1 variables correspond
#  to OBS_VAR1 variables
# Note [FCST/OBS/BOTH]_GRID_STAT_VAR<n>_NAME can be used instead if different evaluations
# are needed for different tools

# Name of forecast variable 1
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

# List of levels to evaluate for forecast variable 1
# A03 = 3 hour accumulation in GRIB file
FCST_VAR1_LEVELS =
```

(continues on next page)

(continued from previous page)

```
# List of thresholds to evaluate for each name/level combination for
#  forecast variable 1
FCST_VAR1_THRESH =

#FCST_GRID_STAT_FILE_TYPE =

# Name of observation variable 1
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


# List of levels to evaluate for observation variable 1
# (*,*) is NetCDF notation - must include quotes around these values!
# must be the same length as FCST_VAR1_LEVELS
OBS_VAR1_LEVELS =

# List of thresholds to evaluate for each name/level combination for
#  observation variable 1
OBS_VAR1_THRESH =

#GRID_STAT_MET_CONFIG_OVERRIDES = cat_thresh = [>=0.15];
#BOTH_VAR1_THRESH = >=0.15

#OBS_GRID_STAT_FILE_TYPE =

# Time relative to valid time (in seconds) to allow files to be considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
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

# MET GridStat neighborhood values
# See the MET User's Guide GridStat section for more information

# width value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_WIDTH = 1

# shape value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE
```

(continues on next page)

(continued from previous page)

```
# cov thresh list passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

# Set to true to run GridStat separately for each field specified
# Set to false to create one run of GridStat per run time that
#   includes all fields specified.
GRID_STAT_ONCE_PER_FIELD = False

# Set to true if forecast data is probabilistic
FCST_IS_PROB = false

# Only used if FCST_IS_PROB is true - sets probabilistic threshold
FCST_GRID_STAT_PROB_THRESH = ==0.1

# Set to true if observation data is probabilistic
#  Only used if configuring forecast data as the 'OBS' input
OBS_IS_PROB = false

# Only used if OBS_IS_PROB is true - sets probabilistic threshold
OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = {custom}

#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 =
#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 =
```

(continues on next page)

(continued from previous page)

```
#GRID_STAT_CLIMO_CDF_BINS = 1
#GRID_STAT_CLIMO_CDF_CENTER_BINS = False
#GRID_STAT_CLIMO_CDF_WRITE_BINS = True

#GRID_STAT_OUTPUT_FLAG_FHO = NONE
#GRID_STAT_OUTPUT_FLAG_CTC = NONE
#GRID_STAT_OUTPUT_FLAG_CTS = NONE
#GRID_STAT_OUTPUT_FLAG_MCTC = NONE
#GRID_STAT_OUTPUT_FLAG_MCTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = BOTH
#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_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


# End of [config] section and start of [dir] section
[dir]
#use case configuration file directory
CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsOSTIA_iceCover
```

(continues on next page)

(continued from previous page)

```
# directory containing forecast input to GridStat
FCST_GRID_STAT_INPUT_DIR =

# directory containing observation input to GridStat
OBS_GRID_STAT_INPUT_DIR =

# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_DIR =

# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_STDEV_INPUT_DIR =

# directory to write output from GridStat
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to GridStat relative to FCST_GRID_STAT_INPUT_DIR
FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to look for observation input to GridStat relative to OBS_GRID_STAT_INPUT_DIR
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

# Optional subdirectories relative to GRID_STAT_OUTPUT_DIR to write output from GridStat
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_MEAN_INPUT_
↪DIR
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_STDEV_INPUT_
↪DIR
# Not used in this example
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

# Used to specify one or more verification mask files for GridStat
# Not used for this example
GRID_STAT_VERIFICATION_MASK_TEMPLATE =
```

5.2.5.2.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.5.2.8 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
```

(continues on next page)

(continued from previous page)

```
from pyproj import Geod
import pyresample as pyr
from datetime import datetime, date
import os, sys

#--------------------------------------
def iceArea(lon1,lat1,ice1):
    """
    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')
```

(continues on next page)

(continued from previous page)

```
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')
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)
```

(continues on next page)

(continued from previous page)

```
# 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')

# 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())
```

(continues on next page)

(continued from previous page)

```
    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",
            '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)
```

(continues on next page)

(continued from previous page)

```
    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': '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
```

5.2.5.2.9 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.

5.2.5.2.10 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

5.2.5.2.11 Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.5.3 Grid-Stat and MODE: Sea Ice Validation

model_applications/marine_and_cryosphere/GridStat_MODE_fcstIMS _obsNCEP_sea_ice.conf

5.2.5.3.1 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.

5.2.5.3.2 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)

5.2.5.3.3 METplus Components

This use case runs the MET GridStat and MODE tools.

5.2.5.3.4 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 901) section for more information).

5.2.5.3.5 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.

```
# IMS Ice Concentration (fcst) vs. NCEP Ice Concentration (obs)
# IMS and NCEP are both observation analyses. For the purpose of running MET, IMS is␣
↪referred to as
# the forecast and NCEP as the obs.
# Written by Lindsay Blank, NCAR. January 2020
###########################################################################################
↪#######
[config]
# Loop by analysis time
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END
VALID_TIME_FMT = %Y%m%d

# L: Available dates are 20190201 - 20190228
# Start time for METplus run
VALID_BEG=20190201

# End time for METplus run
VALID_END=20190201

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT=86400

# Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
# specified, then repeat for the next item in the PROCESS_LIST.
```

(continues on next page)

(continued from previous page)

```
LOOP_ORDER = times

# List of applications to run
PROCESS_LIST = GridStat, Mode

# Description of data to be processed
# used in output file path
MODEL = IMS
OBTYPE = NCEP

# Set to true if forecast data is probabilistic
FCST_IS_PROB = false


###############################################################################
↪######
# GridStat Configurations
###############################################################################
↪######
# List of variables to compare
# "THRESH" refers to "cat_thresh"
FCST_VAR1_NAME = ICEC
FCST_VAR1_LEVELS = Z0
FCST_VAR1_THRESH = ==1.0

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 ];

GRID_STAT_NEIGHBORHOOD_WIDTH = 3, 5, 7, 9

# regridding domain for GridStat
GRID_STAT_REGRID_TO_GRID = OBS

# Location of grid_stat MET config file
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

# prefix to add to GridStat output filenames
GRID_STAT_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}_
↪{CURRENT_FCST_LEVEL}

GRID_STAT_MASK_GRID =
```

(continues on next page)

(continued from previous page)

```
#GRID_STAT_MASK_POLY =

GRID_STAT_OUTPUT_FLAG_CTC = STAT
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 Configurations
###############################################################################################
↪#######
# regridding domain for MODE
MODE_REGRID_TO_GRID = OBS

# Turn on quilting
MODE_QUILT = False

# Convolution radius list
MODE_CONV_RADIUS = 50

# Convolution threshold list
# L: IMS is a binary field where a value of "1" is equivalent to >=0.40 sea ice␣
↪concentration.
FCST_MODE_CONV_THRESH = ==1.00
OBS_MODE_CONV_THRESH = >=0.40

# Location of mode MET config file
MODE_CONFIG_FILE = {PARM_BASE}/met_config/MODEConfig_wrapped

# Merge flag: options are NONE, THRESH, ENGINE, or BOTH
MODE_MERGE_FLAG = NONE

# Merge threshold list
MODE_MERGE_THRESH = >=1.25

MODE_GRID_RES = 12.7

MODE_OBS_CENSOR_THRESH = >1.00 && <=1.28, >1.28
```

(continues on next page)

(continued from previous page)

```
MODE_OBS_CENSOR_VAL = 1.00 , -9999

MODE_MASK_MISSING_FLAG = BOTH

MODE_MATCH_FLAG = NO_MERGE

MODE_MASK_POLY_FLAG = BOTH

MODE_TOTAL_INTEREST_THRESH = 0.8

# prefix to add to MODE output filenames
MODE_OUTPUT_PREFIX = {MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}

[dir]
# input and output data directories for each application in PROCESS_LIST
OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/NCEP_
↪data
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/IMS_
↪data

OBS_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/NCEP_data
FCST_MODE_INPUT_DIR = {INPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/IMS_data

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/
↪GridStat
MODE_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/marine_and_cryosphere/sea_ice/MODE

[filename_templates]
# format of filenames

# IMS
FCST_GRID_STAT_INPUT_TEMPLATE = imssnow96.{valid?fmt=%Y%m%d}.grb.grib2
FCST_MODE_INPUT_TEMPLATE = imssnow96.{valid?fmt=%Y%m%d}.grb.grib2

# NCEP
OBS_GRID_STAT_INPUT_TEMPLATE = seaice.t00z.north12psg.grib2.{valid?fmt=%Y%m%d}
OBS_MODE_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

MODE_VERIFICATION_MASK_TEMPLATE = {INPUT_BASE}/model_applications/marine_and_cryosphere/sea_
↪ice/seaice_nland127.nc

GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/grid_stat
MODE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/mode
```

5.2.5.3.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVER-WRITTEN 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 68)

GridStatConfig_wrapped

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//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 159) 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}

//
```

(continues on next page)

(continued from previous page)

```
// 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 =
${METPLUS_MULTIVAR_LOGIC}

//
// Forecast and observation fields to be verified
//
fcst = {
```

(continues on next page)

(continued from previous page)

```
   ${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}
}

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}
}

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

//
// 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
```

(continues on next page)

(continued from previous page)

```
//
//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 = (
      (  0.00, 1.0 )
      (  0.10, 1.0 )
      (  0.75, 0.0 )
   );

```

(continues on next page)

(continued from previous page)

```
   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
//
met_data_dir = "MET_BASE";

fcst_raw_plot = {
   color_table      = "MET_BASE/colortables/met_default.ctable";
```

(continues on next page)

(continued from previous page)

```
   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}


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

shift_right = 0;   //  grid squares

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

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0";

tmp_dir = "${MET_TMP_DIR}";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.5.3.7 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.

5.2.5.3.8 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

5.2.5.3.9 Keywords

Note:

- GridStatToolUseCase
- MODEToolUseCase
- MarineAndCryosphereAppUseCase
- ValidationUseCase
- S2SAppUseCase
- NOAAEMCOrgUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.5.4 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

5.2.5.4.1 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.

5.2.5.4.2 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 939) section for more information.

Data Source: JPL's PODAAC and NCEP's FTPPRD data servers

5.2.5.4.3 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

5.2.5.4.4 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

5.2.5.4.5 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

5.2.5.4.6 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

```
# GridStat METplus Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# List of applications to run - only GridStat for this case
PROCESS_LIST = GridStat

# time looping - 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
LOOP_BY = VALID

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match INIT_TIME_FMT
VALID_BEG=20210502

# End time for METplus run - must match INIT_TIME_FMT
VALID_END=20210502

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 24


# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
```

(continues on next page)

(continued from previous page)

```
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for GridStat only
LOG_GRID_STAT_VERBOSITY = 2

# Location of MET config file to pass to GridStat
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

# grid to remap data. Value is set as the 'to_grid' variable in the 'regrid' dictionary
# See MET User's Guide for more information
GRID_STAT_REGRID_TO_GRID = NONE

#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 = AREA

# Name to identify model (forecast) data in output
MODEL = RTOFS

# Name to identify observation data in output
OBTYPE = SMAP

# set the desc value in the GridStat MET config file
GRID_STAT_DESC = NA

# List of variables to compare in GridStat - FCST_VAR1 variables correspond
#  to OBS_VAR1 variables
# Note [FCST/OBS/BOTH]_GRID_STAT_VAR<n>_NAME can be used instead if different evaluations
# are needed for different tools

# Name of forecast variable 1
```

(continues on next page)

(continued from previous page)

```
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

# List of levels to evaluate for forecast variable 1
# A03 = 3 hour accumulation in GRIB file
FCST_VAR1_LEVELS =

# List of thresholds to evaluate for each name/level combination for
#  forecast variable 1
FCST_VAR1_THRESH =

#FCST_GRID_STAT_FILE_TYPE =

# Name of observation variable 1
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


# List of levels to evaluate for observation variable 1
# (*,*) is NetCDF notation - must include quotes around these values!
# must be the same length as FCST_VAR1_LEVELS
OBS_VAR1_LEVELS =

# List of thresholds to evaluate for each name/level combination for
#  observation variable 1
OBS_VAR1_THRESH =

#GRID_STAT_MET_CONFIG_OVERRIDES = cat_thresh = [>=0.15];
#BOTH_VAR1_THRESH = >=0.15

#OBS_GRID_STAT_FILE_TYPE =



# Name of climatology variable 1
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="(*,*)";}
```

(continues on next page)

(continued from previous page)

```
# Time relative to valid time (in seconds) to allow files to be considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
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

# MET GridStat neighborhood values
# See the MET User's Guide GridStat section for more information

# width value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_WIDTH = 1

# shape value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

# cov thresh list passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

# Set to true to run GridStat separately for each field specified
# Set to false to create one run of GridStat per run time that
#   includes all fields specified.
GRID_STAT_ONCE_PER_FIELD = False

# Set to true if forecast data is probabilistic
FCST_IS_PROB = false

# Only used if FCST_IS_PROB is true - sets probabilistic threshold
FCST_GRID_STAT_PROB_THRESH = ==0.1

# Set to true if observation data is probabilistic
#  Only used if configuring forecast data as the 'OBS' input
OBS_IS_PROB = false

# Only used if OBS_IS_PROB is true - sets probabilistic threshold
OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = SSS

#GRID_STAT_CLIMO_MEAN_FILE_NAME =
#GRID_STAT_CLIMO_MEAN_FIELD =
#GRID_STAT_CLIMO_MEAN_REGRID_METHOD =
```

(continues on next page)

(continued from previous page)

```
#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 =
#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_OUTPUT_FLAG_FHO = NONE
#GRID_STAT_OUTPUT_FLAG_CTC = NONE
#GRID_STAT_OUTPUT_FLAG_CTS = NONE
#GRID_STAT_OUTPUT_FLAG_MCTC = NONE
#GRID_STAT_OUTPUT_FLAG_MCTS = NONE
GRID_STAT_OUTPUT_FLAG_CNT = BOTH
#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
```

(continues on next page)

(continued from previous page)

```
#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


# End of [config] section and start of [dir] section
[dir]
#use case configuration file directory
CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsSMAP_climWOA_sss
# directory containing forecast input to GridStat
FCST_GRID_STAT_INPUT_DIR =

# directory containing observation input to GridStat
OBS_GRID_STAT_INPUT_DIR =

# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_DIR =

# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_STDEV_INPUT_DIR =

# directory to write output from GridStat
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to GridStat relative to FCST_GRID_STAT_INPUT_DIR
FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to look for observation input to GridStat relative to OBS_GRID_STAT_INPUT_DIR
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY
```

(continues on next page)

(continued from previous page)

```
# Optional subdirectories relative to GRID_STAT_OUTPUT_DIR to write output from GridStat
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_MEAN_INPUT_
↪DIR
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_STDEV_INPUT_
↪DIR
# Not used in this exampls
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

# Used to specify one or more verification mask files for GridStat
# Not used for this example
GRID_STAT_VERIFICATION_MASK_TEMPLATE =
```

5.2.5.4.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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
//
```

(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;
   rep_prop = 1.0;
```

(continues on next page)

(continued from previous page)

```
   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
//
gradient = {
```

(continues on next page)

(continued from previous page)

```
   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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.5.4.8 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
"""

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'
```

(continues on next page)

(continued from previous page)

```
###################################################################
# 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'})


# 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')
```

(continues on next page)

(continued from previous page)

```
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,]

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
```

(continues on next page)

(continued from previous page)

```
######################################################################
# 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')
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
```

(continues on next page)

(continued from previous page)

```
    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
    """

    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
```

(continues on next page)

(continued from previous page)

```
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[:,:]
    #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,
```

(continues on next page)

(continued from previous page)

```
                '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")
    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,
                }
```

(continues on next page)

(continued from previous page)

```
            }
    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]
    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
```

5.2.5.4.9 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.

5.2.5.4.10 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

5.2.5.4.11 Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.5.5 GridStat: Python Embedding to read and process SST

model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsGHRSST_climWOA_sst.conf

5.2.5.5.1 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.

5.2.5.5.2 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 957) section for more information.

Data Source: JPL's PODAAC and NCEP's FTPPRD data servers

5.2.5.5.3 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

5.2.5.5.4 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.

5.2.5.5.5 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

5.2.5.5.6 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]

PROCESS_LIST = GridStat

###
```

(continues on next page)

(continued from previous page)

```
# Time Info
###

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG=20210503
VALID_END=20210503
VALID_INCREMENT = 1M

LEAD_SEQ = 0

LOOP_ORDER = times

###
# Field Info
###

MODEL = RTOFS
OBTYPE = GHRSST

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsGHRSST_climWOA_sst

FCST_IS_PROB = false

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_
↪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
FCST_VAR1_LEVELS =
FCST_VAR1_THRESH =

OBS_IS_PROB = false

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
OBS_VAR1_LEVELS =
```

(continues on next page)

(continued from previous page)

```
OBS_VAR1_THRESH =

###
# File I/O
###

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 = PYTHON_NUMPY

GRID_STAT_CLIMO_STDEV_INPUT_DIR =
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}

###
# GridStat
###

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

GRID_STAT_REGRID_TO_GRID = NONE

GRID_STAT_DESC = NA

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="(*,*)
↪";}

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)

```
GRID_STAT_NEIGHBORHOOD_WIDTH = 1
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

GRID_STAT_ONCE_PER_FIELD = False

FCST_GRID_STAT_PROB_THRESH = ==0.1

OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = SST

GRID_STAT_OUTPUT_FLAG_CNT = BOTH
```

5.2.5.5.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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 = {
  ${METPLUS_FCST_FILE_TYPE}
```

(continues on next page)

(continued from previous page)

```
  ${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;
   rng      = "mt19937";
```

(continues on next page)

(continued from previous page)

```
   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 ];
   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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.5.5.8 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
"""

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'
```

(continues on next page)

(continued from previous page)

```
###################################################################
# 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)

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')
```

(continues on next page)

(continued from previous page)

```
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)
#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
```

(continues on next page)

(continued from previous page)

```
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')
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)
```

(continues on next page)

(continued from previous page)

```
    #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
↪'])

    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()
```

(continues on next page)

(continued from previous page)

```
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()

#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',
```

(continues on next page)

(continued from previous page)

```
            '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':
    #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",
```

(continues on next page)

(continued from previous page)

```
            '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':
    #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",
```

(continues on next page)

(continued from previous page)

```
            '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
```

5.2.5.5.9 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
```

5.2.5.5.10 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

5.2.5.5.11 Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.5.6 PlotDataPlane: Python Embedding of tripolar coordinate file

model_applications/marine_and_cryosphere/PlotDataPlane_obsHYCOM_coordTripolar.conf

5.2.5.6.1 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.

5.2.5.6.2 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 966) section for more information.

Data Source: HYCOM model

5.2.5.6.3 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

5.2.5.6.4 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.

5.2.5.6.5 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.

5.2.5.6.6 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]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - only PlotDataPlane for this case
PROCESS_LIST = PlotDataPlane

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 20200127

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 20200127
```

(continues on next page)

(continued from previous page)

```
# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 0

PLOT_DATA_PLANE_CUSTOM_LOOP_LIST = north, south

LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for PlotDataPlane only
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 =


# End of [config] section and start of [dir] section
[dir]

# Input/Output directories can be left empty if the corresponding template contains the full␣
↪path to the files
PLOT_DATA_PLANE_INPUT_DIR =
PLOT_DATA_PLANE_OUTPUT_DIR =

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for input to PlotDataPlane relative to PLOT_DATA_PLANE_INPUT_DIR
PLOT_DATA_PLANE_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to use to write output from PlotDataPlane
PLOT_DATA_PLANE_OUTPUT_TEMPLATE = {OUTPUT_BASE}/model_applications/marine_and_cryosphere/
↪PlotDataPlane_obsHYCOM_coordTripolar/HYCOM_iceCoverage_{custom}.ps
```

5.2.5.6.7 MET Configuration

This tool does not use a MET configuration file.

5.2.5.6.8 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))
```

5.2.5.6.9 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.

5.2.5.6.10 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

5.2.5.6.11 Keywords

Note:

- PlotDataPlaneToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.5.7 GridStat: Python Embedding to read and process sea surface heights

model_applications/marine_and_cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh.conf

5.2.5.7.1 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.

5.2.5.7.2 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 987) section for more information.

Data Source: COPERNICUS GLOBAL OCEAN SSH NRT (LEVEL 4), HYCOM + NCODA Global 1/12 deg Reanalysis

5.2.5.7.3 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

5.2.5.7.4 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

5.2.5.7.5 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

5.2.5.7.6 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

```
# GridStat METplus Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# List of applications to run - only GridStat for this case
PROCESS_LIST = GridStat

# time looping - 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
LOOP_BY = VALID

# Format of INIT_BEG and INT_END using % items
```

(continues on next page)

(continued from previous page)

```
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match INIT_TIME_FMT
VALID_BEG=20210811

# End time for METplus run - must match INIT_TIME_FMT
VALID_END=20210811

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 1M

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 24


# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for GridStat only
LOG_GRID_STAT_VERBOSITY = 2

# Location of MET config file to pass to GridStat
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

# grid to remap data. Value is set as the 'to_grid' variable in the 'regrid' dictionary
# See MET User's Guide for more information
GRID_STAT_REGRID_TO_GRID = NONE

# Name to identify model (forecast) data in output
MODEL = RTOFS

# Name to identify observation data in output
```

(continues on next page)

(continued from previous page)

```
OBTYPE = AVISO

# set the desc value in the GridStat MET config file
GRID_STAT_DESC = NA

# List of variables to compare in GridStat - FCST_VAR1 variables correspond
#  to OBS_VAR1 variables
# Note [FCST/OBS/BOTH]_GRID_STAT_VAR<n>_NAME can be used instead if different evaluations
# are needed for different tools

# Name of forecast variable 1
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

# List of levels to evaluate for forecast variable 1
# A03 = 3 hour accumulation in GRIB file
FCST_VAR1_LEVELS =

# List of thresholds to evaluate for each name/level combination for
#  forecast variable 1
FCST_VAR1_THRESH =

#FCST_GRID_STAT_FILE_TYPE =

# Name of observation variable 1
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_
↪cryosphere/GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh {valid?fmt=%Y%m%d} obs


# List of levels to evaluate for observation variable 1
# (*,*) is NetCDF notation - must include quotes around these values!
# must be the same length as FCST_VAR1_LEVELS
OBS_VAR1_LEVELS =

# List of thresholds to evaluate for each name/level combination for
```

(continues on next page)

(continued from previous page)

```
#  observation variable 1
OBS_VAR1_THRESH =

#GRID_STAT_MET_CONFIG_OVERRIDES = cat_thresh = [>=0.15];
#BOTH_VAR1_THRESH = >=0.15

#OBS_GRID_STAT_FILE_TYPE =


# Name of climatology variable 1
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="(*,*)";}


# Time relative to valid time (in seconds) to allow files to be considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
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

# MET GridStat neighborhood values
# See the MET User's Guide GridStat section for more information

# width value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_WIDTH = 1

# shape value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

# cov thresh list passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

# Set to true to run GridStat separately for each field specified
# Set to false to create one run of GridStat per run time that
#   includes all fields specified.
GRID_STAT_ONCE_PER_FIELD = False
```

(continues on next page)

(continued from previous page)

```
# Set to true if forecast data is probabilistic
FCST_IS_PROB = false

# Only used if FCST_IS_PROB is true - sets probabilistic threshold
FCST_GRID_STAT_PROB_THRESH = ==0.1

# Set to true if observation data is probabilistic
#  Only used if configuring forecast data as the 'OBS' input
OBS_IS_PROB = false

# Only used if OBS_IS_PROB is true - sets probabilistic threshold
OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = SSH

GRID_STAT_OUTPUT_FLAG_CNT = BOTH
GRID_STAT_OUTPUT_FLAG_SAL1L2 = BOTH


# End of [config] section and start of [dir] section
[dir]
#use case configuration file directory
CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/marine_and_cryosphere/GridStat_
↪fcstRTOFS_obsAVISO_climHYCOM_ssh
# directory containing forecast input to GridStat
FCST_GRID_STAT_INPUT_DIR =

# directory containing observation input to GridStat
OBS_GRID_STAT_INPUT_DIR =

# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_DIR =

# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_STDEV_INPUT_DIR =

# directory to write output from GridStat
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to GridStat relative to FCST_GRID_STAT_INPUT_DIR
```

(continues on next page)

(continued from previous page)

```
FCST_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to look for observation input to GridStat relative to OBS_GRID_STAT_INPUT_DIR
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY

# Optional subdirectories relative to GRID_STAT_OUTPUT_DIR to write output from GridStat
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_MEAN_INPUT_
↪DIR
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE = PYTHON_NUMPY

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_STDEV_INPUT_
↪DIR
# Not used in this exampls
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

# Used to specify one or more verification mask files for GridStat
# Not used for this example
GRID_STAT_VERIFICATION_MASK_TEMPLATE =
```

5.2.5.7.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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 =
${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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.5.7.8 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)

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'
```

(continues on next page)

(continued from previous page)

```
######################################################################
# 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'

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)
```

(continues on next page)

(continued from previous page)

```
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)
    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)
```

(continues on next page)

(continued from previous page)

```
    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')
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()
```

(continues on next page)

(continued from previous page)

```
    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):
    """
    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')
```

(continues on next page)

(continued from previous page)

```
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)

#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
```

(continues on next page)

(continued from previous page)

```
    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,
                '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'
```

(continues on next page)

(continued from previous page)

```
    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,
                }
            }
    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]
```

(continues on next page)

(continued from previous page)

```
    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
```

5.2.5.7.9 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.

5.2.5.7.10 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

5.2.5.7.11 Keywords

Note:

- GridStatToolUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/marine_and_cryosphere-GridStat_fcstRTOFS_obsAVISO_climHYCOM_ssh.p

Total running time of the script: (0 minutes 0.000 seconds)

5.2.5.8 UserScript: Python Script to compute cable transport

model_applications/marine_and_cryosphere/UserScript_fcstRTOFS_obsAOML_calcTransport.conf

5.2.5.8.1 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.

5.2.5.8.2 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 997) 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)

5.2.5.8.3 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
```

5.2.5.8.4 METplus Components

This use case utilizes the METplus UserScript wrapper to generate a command to run with Python Embedding for the specified valid time.

5.2.5.8.5 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.

5.2.5.8.6 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]

# List of applications to run
PROCESS_LIST = UserScript

# time looping - 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
LOOP_BY = VALID

# Format of INIT_BEG and INIT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 20211028

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 24H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ =

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run

LOOP_ORDER = processes

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE

USER_SCRIPT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/marine_and_cryosphere/calc_
↪transport

USER_SCRIPT_INPUT_TEMPLATE = {VALID_BEG}

# The valid date increments backwards
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}
```

(continues on next page)

(continued from previous page)

```
[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 = {OUTPUT_BASE}/model_applications/marine_and_cryosphere/calc_transport
```

5.2.5.8.7 MET Configuration

None. All of the processing is completed in the UserScript

5.2.5.8.8 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
```

(continues on next page)

(continued from previous page)

```
"""

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=' ',
    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.
```

(continues on next page)

(continued from previous page)

```
    """
    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]))
        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:])
```

(continues on next page)

(continued from previous page)

```
        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
#-----------------------------------------------
# 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
```

(continues on next page)

(continued from previous page)

```
    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
    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)
```

(continues on next page)

(continued from previous page)

```
    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)
```

5.2.5.8.9 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
```

5.2.5.8.10 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

5.2.5.8.11 Keywords

Note:

- UserScriptUseCase
- PythonEmbeddingFileUseCase
- MarineAndCryosphereAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.6 Medium Range

Lower resolution model configuration (>4km) usually producing forecasts out to 7-14 days (also referred to as global models)

5.2.6.1 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

5.2.6.1.1 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.

5.2.6.1.2 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

5.2.6.1.3 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
```

5.2.6.1.4 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.

5.2.6.1.5 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

5.2.6.1.6 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

```
#
#  CONFIGURATION
#
[config]

# Loop over each process in the process list (set in PROCESS_LIST) for all times in the time␣
↪window of interest.
LOOP_ORDER = processes

#Configuration files
TC_PAIRS_CONFIG_FILE = {PARM_BASE}/met_config/TCPairsConfig_wrapped

SERIES_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/SeriesAnalysisConfig_wrapped

PROCESS_LIST = PyEmbedIngest, TCPairs, TCStat, ExtractTiles, TCStat(for_series_analysis),␣
↪SeriesAnalysis

# The init time begin and end times, increment
# Looping - options are INIT, VALID, RETRO, and REALTIME
```

(continues on next page)

(continued from previous page)

```
LOOP_BY = INIT

# Format of INIT_BEG and INIT_END
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run
INIT_BEG = 2019083000

# End time for METplus run
INIT_END = 2019083023

# This is the step-size. Increment in seconds from the begin time to the end time
INIT_INCREMENT = 21600 ;; set to every 6 hours=21600 seconds

# Used by extract tiles and series analysis to define the records of
#  interest to be retrieved from the grib2 file

BOTH_VAR1_NAME = ivt
BOTH_VAR1_LEVELS = Surface

BOTH_VAR2_NAME = pv
BOTH_VAR2_LEVELS = Surface

BOTH_VAR3_NAME = sept
BOTH_VAR3_LEVELS = Surface

LEAD_SEQ = 90, 96, 102, 108, 114

#####
### PYEMBED INGEST
#####

# 1st INGEST INSTANCE: Forecast
# python script with optional arguments to run for 1st ingest instance
# 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
# output variable name
PY_EMBED_INGEST_1_OUTPUT_FIELD_NAME_1 = ivt

# PV
# python script with optional arguments to run for 1st ingest instance
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
# output variable name
```

(continues on next page)

(continued from previous page)

```
PY_EMBED_INGEST_1_OUTPUT_FIELD_NAME_2 = pv

# SEPT
# python script with optional arguments to run for 1st ingest instance
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
# output variable name
PY_EMBED_INGEST_1_OUTPUT_FIELD_NAME_3 = sept

# type of python input to expect for 1st ingest instance
# valid options: NUMPY, XARRAY
PY_EMBED_INGEST_1_TYPE = NUMPY

# output grid for 1st ingest instance. Can be a grid definition or file path
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
# python script with optional arguments to run for 2nd ingest instance
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
# output variable name
PY_EMBED_INGEST_2_OUTPUT_FIELD_NAME_1 = ivt

# PV
# python script with optional arguments to run for 2nd ingest instance
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
# output variable name
PY_EMBED_INGEST_2_OUTPUT_FIELD_NAME_2 = pv

# SEPT
# python script with optional arguments to run for 2nd ingest instance
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
# output variable name
PY_EMBED_INGEST_2_OUTPUT_FIELD_NAME_3 = sept

# type of python input to expect for 2nd ingest instance
# valid options: NUMPY, XARRAY
PY_EMBED_INGEST_2_TYPE = NUMPY

# output grid for 2nd ingest instance. Can be a grid definition or file path
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
```

(continues on next page)

(continued from previous page)

```
#######
# TCPairs Configurations
#######

# only run TCPairs once for the init time, not for each forecast lead
TC_PAIRS_SKIP_LEAD_SEQ = True

# A list of times to include, in format YYYYMMDD_hh
TC_PAIRS_INIT_INCLUDE =

# A list of times to exclude, in format YYYYMMDD_hh
TC_PAIRS_INIT_EXCLUDE =

# Specify model valid time window in format YYYYMM[DD[_hh]].  Only tracks
# that fall within the valid time window will be used.
TC_PAIRS_VALID_BEG =
TC_PAIRS_VALID_END =

#
# Run MET tc_pairs by indicating the top-level directories for the A-deck
# and B-deck files. Set to 'yes' to run using top-level directories, 'no'
# if you want to run tc_pairs on files paired by the wrapper.
TC_PAIRS_READ_ALL_FILES = no

# List of models to be used (white space or comma separated) eg: DSHP, LGEM, HWRF
# If no models are listed, then process all models in the input file(s).
MODEL = GFSO

# List of storm ids of interest (space or comma separated) e.g.: AL112012, AL122012
# If no storm ids are listed, then process all storm ids in the input file(s).
TC_PAIRS_STORM_ID =

# Basins (of origin/region).  Indicate with space or comma-separated list of regions, eg.
↪AL: for North Atlantic,
# WP: Western North Pacific, CP: Central North Pacific, SH: Southern Hemisphere, IO: North
↪Indian Ocean, LS: Southern
# Hemisphere
TC_PAIRS_BASIN =

# Cyclone, a space or comma-separated list of cyclone numbers. If left empty, all cyclones
↪will be used.
TC_PAIRS_CYCLONE =

# Storm name, a space or comma-separated list of storm names to evaluate.  If left empty,
↪all storms will be used.
```

(continues on next page)

(continued from previous page)

```
TC_PAIRS_STORM_NAME =

# DLAND file, the full path of the file that contains the gridded representation of the
# minimum distance from land.
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

# overwrite modified track data (non-ATCF to ATCF format)
TC_PAIRS_SKIP_IF_REFORMAT_EXISTS = no

# overwrite tc_pairs output
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = no

#######
# TCStat Configurations
#######

# IMPORTANT  Refer to the README_TC for details on setting up analysis
# jobs (located in {MET_INSTALL_DIR}/share/met/config

# Separate each option and value with whitespace, and each job with a whitespace.
# No whitespace within arithmetic expressions or lists of items
# (e.g. -by AMSLP,AMODEL,LEAD -column '(AMAX_WIND-BMAX_WIND)')
# Enclose your arithmetic expressions with '' and separate each job
# by whitespace:
#  -job filter -dump_row /path/to,  -job summary -line_type TCMPR  -column 'ABS(AMAX_WIND-
↪BMAX_WIND)' -out {OUTPUT_BASE}/tc_stat/file.tcst

TC_STAT_JOB_ARGS = -job filter -basin AL -dump_row {TC_STAT_OUTPUT_DIR}/{TC_STAT_DUMP_ROW_
↪TEMPLATE}


# Specify whether only those track points common to both the ADECK and BDECK
# tracks should be written out.  This is only used when explicitly calling
# TC_STAT in the PROCESS_LIST.  This is not used in this use case, so setting
# it to either false or true has no impact.
TC_STAT_MATCH_POINTS = true

# Stratify by these columns:
TC_STAT_AMODEL = {MODEL}
```

(continues on next page)

(continued from previous page)

```
TC_STAT_BMODEL =
TC_STAT_DESC =
TC_STAT_STORM_ID =
TC_STAT_BASIN =
TC_STAT_CYCLONE =
TC_STAT_STORM_NAME =

# Stratify by init times via a comma-separate list of init times to
# include or exclude.  Time format defined as YYYYMMDD_HH or YYYYMMDD_HHmmss
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 =
# Stratify by valid times via a comma-separate list of valid times to
# include or exclude.  Time format defined as YYYYMMDD_HH or YYYYMMDD_HHmmss
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 =
# Stratify by the valid time and lead time via comma-separated list of
# times in format HH[MMSS]
TC_STAT_VALID_HOUR =
TC_STAT_LEAD =

# Stratify over the watch_warn column in the tcst file.  Setting this to
# 'ALL' will match HUWARN, HUWATCH, TSWARN, TSWATCH
TC_STAT_TRACK_WATCH_WARN =

# Stratify by applying thresholds to numeric data columns.  Specify with
# comma-separated list of column names and thresholds to be applied.
# The length of TC_STAT_COLUMN_THRESH_NAME should be the same as
# TC_STAT_COLUMN_THRESH_VAL.
TC_STAT_COLUMN_THRESH_NAME =
TC_STAT_COLUMN_THRESH_VAL =

# Stratify by a list of comma-separated columns names and values corresponding
# to non-numeric data columns of the values of interest.
TC_STAT_COLUMN_STR_NAME =
TC_STAT_COLUMN_STR_VAL =
```

(continues on next page)

(continued from previous page)

```
# Stratify by applying thresholds to numeric data columns only when lead=0.
# If lead=0 and the value does not meet the threshold, discard the entire
# track.  The length of TC_STAT_INIT_THRESH_NAME must equal the length of
# TC_STAT_INIT_THRESH_VAL.
TC_STAT_INIT_THRESH_NAME =
TC_STAT_INIT_THRESH_VAL =

# Stratify by applying thresholds to numeric data columns only when lead = 0.
# If lead = 0 but the value doesn't meet the threshold, discard the entire
# track.
TC_STAT_INIT_STR_NAME =
TC_STAT_INIT_STR_VAL =

# Excludes any points where distance to land is <=0. When 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.
TC_STAT_WATER_ONLY =

# TRUE or FALSE.  To specify whether only those track points occurring near
# landfall should be retained. Landfall is the last bmodel track point before
# the distance to land switches from water to land.
TC_STAT_LANDFALL =

# Define the landfall retention window, which is defined as the hours offset
# from the time of landfall. Format is in HH[MMSS]. Default TC_STAT_LANDFALL_BEG
# is set to -24, and TC_STAT_LANDFALL_END is set to 00
TC_STAT_LANDFALL_BEG =
TC_STAT_LANDFALL_END =


# Constants used in creating the tile grid, used by extract tiles
EXTRACT_TILES_NLAT = 60
EXTRACT_TILES_NLON = 60

# Resolution of data in degrees, used by extract tiles
EXTRACT_TILES_DLAT = 0.5
EXTRACT_TILES_DLON = 0.5

# Degrees to subtract from the center lat and lon to
# calculate the lower left lat (lat_ll) and lower
# left lon (lon_ll) for a grid that is 2n X 2m,
# where n = EXTRACT_TILES_LAT_ADJ degrees and m = EXTRACT_TILES_LON_ADJ degrees.
# For this case, where n=15 and m=15, this results
# in a 30 deg X 30 deg grid.  Used by extract tiles
EXTRACT_TILES_LON_ADJ = 15
```

(continues on next page)

(continued from previous page)

```
EXTRACT_TILES_LAT_ADJ = 15

# OVERWRITE OPTIONS
# Skip writing filter files if they already exist.
# Set to yes if you want to skip processing existing files
# Set to no if you want to override existing files
EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS = no

# 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}

[config]

# set the regrid dictionary item to_grid in the SeriesAnalysis MET config file
SERIES_ANALYSIS_REGRID_TO_GRID = FCST
SERIES_ANALYSIS_REGRID_METHOD = FORCE
#SERIES_ANALYSIS_REGRID_WIDTH =
#SERIES_ANALYSIS_REGRID_VLD_THRESH =
#SERIES_ANALYSIS_REGRID_SHAPE =

# NOTE: "TOTAL" is a REQUIRED cnt statistic used by the series analysis scripts
SERIES_ANALYSIS_STAT_LIST = TOTAL, FBAR, OBAR, ME

# PLOTTING Relevant to series analysis plots.
# By default, background map is turned off. Set
# to no to turn of plotting of background map.
SERIES_ANALYSIS_BACKGROUND_MAP = yes

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False

SERIES_ANALYSIS_BLOCK_SIZE = 4000

# set to True to add the -paired flag to the SeriesAnalysis command
SERIES_ANALYSIS_IS_PAIRED = True

# If True/yes, run plot_data_plane on output from Series-Analysis to generate
```

(continues on next page)

(continued from previous page)

```
# images for each stat item listed in SERIES_ANALYSIS_STAT_LIST
SERIES_ANALYSIS_GENERATE_PLOTS = yes

# If True/yes, run convert on output from Series-Analysis to generate
# a gif using images in groups of name/level/stat
SERIES_ANALYSIS_GENERATE_ANIMATIONS = yes

PLOT_DATA_PLANE_TITLE = {MODEL} series_F{fcst_beg} Forecasts{nseries}, {stat} for {fcst_name}
↪ {fcst_level}

#
#  FILENAME TEMPLATES
#
[filename_templates]
# Define the format of the filenames
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_TEMPLATE = {valid?fmt=%Y%m%d}/gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=
↪%H}00_000.nc

TC_PAIRS_ADECK_TEMPLATE = a{basin?fmt=%s}052019.dat
TC_PAIRS_BDECK_TEMPLATE = b{basin?fmt=%s}052019.dat
TC_PAIRS_OUTPUT_TEMPLATE = {date?fmt=%Y%m}/{basin?fmt=%s}q{date?fmt=%Y%m%d%H}.dorian

TC_STAT_DUMP_ROW_TEMPLATE = filter_{init?fmt=%Y%m%d_%H}.tcst

EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}
FCST_EXTRACT_TILES_INPUT_TEMPLATE = {init?fmt=%Y%m%d}/gfs_4_{init?fmt=%Y%m%d}_{init?fmt=%H}
↪00_{lead?fmt=%3H}.nc
OBS_EXTRACT_TILES_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}
↪00_000.nc

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

FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = {FCST_EXTRACT_TILES_OUTPUT_TEMPLATE}
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = {OBS_EXTRACT_TILES_OUTPUT_TEMPLATE}

SERIES_ANALYSIS_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}


# Template to look for climatology mean input to SeriesAnalysis relative to SERIES_ANALYSIS_
↪CLIMO_MEAN_INPUT_DIR
```

(continues on next page)

(continued from previous page)

```
# Not used in this example
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_TEMPLATE =

# Template to look for climatology standard deviation input to SeriesAnalysis relative to␣
↪SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR
# Not used in this example
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_TEMPLATE =

SERIES_ANALYSIS_OUTPUT_TEMPLATE = {label}/series_F{fcst_beg}_{fcst_name}_{fcst_level}.nc
#
#  DIRECTORIES
#
[dir]

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/use_cases/model_applications/medium_range/TCStat_SeriesAnalysis_
↪fcstGFS_obsGFS_FeatureRelative_SeriesByLead_PyEmbed_Multiple_Diagnostics

#Location of model data
MODEL_DIR = {INPUT_BASE}/model_applications/medium_range/dorian_data/model_data


PY_EMBED_INGEST_1_OUTPUT_DIR = {OUTPUT_BASE}/py_embed_out
PY_EMBED_INGEST_2_OUTPUT_DIR = {PY_EMBED_INGEST_1_OUTPUT_DIR}

# track data, set to your data source
TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/dorian_data/track_
↪data
TC_PAIRS_BDECK_INPUT_DIR = {TC_PAIRS_ADECK_INPUT_DIR}
TC_PAIRS_REFORMAT_DIR = {OUTPUT_BASE}/track_data_atcf
TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs

TC_STAT_LOOKIN_DIR = {TC_PAIRS_OUTPUT_DIR}
TC_STAT_OUTPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}

EXTRACT_TILES_TC_STAT_INPUT_DIR = {TC_STAT_OUTPUT_DIR}
EXTRACT_TILES_GRID_INPUT_DIR = {PY_EMBED_INGEST_1_OUTPUT_DIR}
FCST_EXTRACT_TILES_INPUT_DIR = {PY_EMBED_INGEST_1_OUTPUT_DIR}
OBS_EXTRACT_TILES_INPUT_DIR = {PY_EMBED_INGEST_1_OUTPUT_DIR}
EXTRACT_TILES_OUTPUT_DIR = {OUTPUT_BASE}/extract_tiles

FCST_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
OBS_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
SERIES_ANALYSIS_TC_STAT_INPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}
```

(continues on next page)

(continued from previous page)

```
# directory containing climatology mean input to SeriesAnalysis
# Not used in this example
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_DIR =

# directory containing climatology standard deviation input to SeriesAnalysis
# Not used in this example
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR =

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/series_analysis_lead

[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
```

5.2.6.1.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) for more details.

YOU SHOULD NOT SET ANY OF THESE ENVIRONMENT VARIABLES YOURSELF! THEY WILL BE OVER-WRITTEN 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 68)

TCPairsConfig_wrapped

Note: See the *TCPairs MET Configuration* (page 243) 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
//
```

(continues on next page)

(continued from previous page)

```
//
// 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}
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
${METPLUS_VALID_INC}
// valid_exc =
${METPLUS_VALID_EXC}
```

(continues on next page)

(continued from previous page)

```
// 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.
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
//
//consensus =
${METPLUS_CONSENSUS_LIST}
```

(continues on next page)

(continued from previous page)

```
//
// 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 = TRUE;

//
// 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;
}

//
```

(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 = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

TCStatConfig_wrapped

Note: See the *TCStat MET Configuration* (page 256) 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.
```

(continues on next page)

(continued from previous page)

```
//
${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_INCLUDE}
${METPLUS_INIT_EXCLUDE}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INCLUDE}
${METPLUS_VALID_EXCLUDE}

//
// Stratify by the initialization and valid hours and lead time.
//
${METPLUS_INIT_HOUR}
${METPLUS_VALID_HOUR}
```

(continues on next page)

(continued from previous page)

```
${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 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}
```

(continues on next page)

(continued from previous page)

```
${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}

//
// 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}

//
// 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 207) 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 =
```

(continues on next page)

(continued from previous page)

```
${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
//
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
```

(continues on next page)

(continued from previous page)

```
   seed     = "";
}

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

//
// Verification masking regions
//
mask = {
   grid = "";
   poly = "";
}

//
// 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";

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

(continues on next page)

(continued from previous page)

```
${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.6.1.8 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
    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))]))
```

(continues on next page)

(continued from previous page)

```
    # 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
        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)
```

(continues on next page)

(continued from previous page)

```
    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)
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)),
```

(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 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 pv(input_file):

    # Vars
    grib_vars = ['t','u','v']

    # Load a list of datasets, one for each variable we want
```

(continues on next page)

(continued from previous page)

```
    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.
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}")
```

(continues on next page)

(continued from previous page)

```
    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
```

(continues on next page)

(continued from previous page)

```
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

###########################################################################################
↪######

input_file = os.path.expandvars(sys.argv[1])
```

(continues on next page)

(continued from previous page)

```
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',

        'grid': {
            'name': 'Global 0.5 Degree',
            'type' :   'LatLon',
            'lat_ll' : -90.0,
            'lon_ll' : 0.0,
            'delta_lat' :   0.5,
```

(continues on next page)

(continued from previous page)

```
            '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

    # 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.
```

(continues on next page)

(continued from previous page)

```
    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
        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
```

(continues on next page)

(continued from previous page)

```
    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',
   'accum': '00',

   'name':      'ivt',
   'long_name': 'integrated_vapor_transport',
   'level':     'Surface',
   'units':     'UNKNOWN',

   'grid': {
```

(continues on next page)

(continued from previous page)

```
        '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']

    # 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]
```

(continues on next page)

(continued from previous page)

```
    # 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}$", token)):
        hh  = int(token[0:2])
    elif(re.search("^[0-9]{3}$", token)):
        day = int(token.replace("", ""))


print("Data Shape: " + repr(met_data.shape))
print("Data Type:  " + repr(met_data.dtype))
```

(continues on next page)

(continued from previous page)

```
# 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
###########################################################################################
↪######

import sys
import os
import re
import datetime as dt
from metpy import calc as mpcalc
```

(continues on next page)

(continued from previous page)

```
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

###########################################################################################
↪######

input_file = os.path.expandvars(sys.argv[1])

data = sept(input_file) #Call function to calculate PV

met_data = data
```

(continues on next page)

(continued from previous page)

```
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',

    'grid': {
        'name': 'Global 0.5 Degree',
        'type' :   'LatLon',
        'lat_ll' : -90.0,
        'lon_ll' : 0.0,
        'delta_lat' :   0.5,
        'delta_lon' :   0.5,
```

(continues on next page)

(continued from previous page)

```
        'Nlat' :      361,
        'Nlon' :      720,
    }
}
```

5.2.6.1.9 Running METplus

This use case can be run two ways:

1) Passing in TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByLead_PyEmbed_Multiple_Diagnostics.conf 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

```

(continues on next page)

(continued from previous page)

```
[exe]
CONVERT = /path/to/convert
```

5.2.6.1.10 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.

5.2.6.1.11 Keywords

Note:

- TCPairsToolUseCase
- SeriesByLeadUseCase
- TCStatToolUseCase
- RegridDataPlaneToolUseCase
- PyEmbedIngestToolUseCase
- MediumRangeAppUseCase
- SeriesAnalysisUseCase
- GRIB2FileUseCase
- FeatureRelativeUseCase
- SBUOrgUseCase
- DiagnosticsUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.6.2 UserScript: Calculate the Difficulty Index

model_applications/medium_range/ UserScript_fcstGEFS _Difficulty_Index.conf

5.2.6.2.1 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.

5.2.6.2.2 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

5.2.6.2.3 METplus Components

This use case runs the UserScript wrapper tool to run a user provided script, in this case, wind_difficulty_index.py.

5.2.6.2.4 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

5.2.6.2.5 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]

# List of applications to run
PROCESS_LIST = UserScript

# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INIT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
INIT_BEG = 2020120812

# End time for METplus run - must match VALID_TIME_FMT
INIT_END = 2020120812

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 12H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ =

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
```

(continues on next page)

(continued from previous page)

```
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# list of strings to loop over for each run time.
# value for each item can be referenced in filename templates with {custom?fmt=%s}
USER_SCRIPT_CUSTOM_LOOP_LIST = nc

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_FOR_EACH

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

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}

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
```

(continues on next page)

(continued from previous page)

```
DIFF_INDEX_FIG_FMT = png

DIFF_INDEX_FIG_BASENAME = {USER_SCRIPT_OUTPUT_DIR}/wndspd_GEFS_NorthPac_5dy_30mem_difficulty_
↪index
```

5.2.6.2.6 MET Configuration

There are no MET tools used in this use case.

5.2.6.2.7 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
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']))
```

(continues on next page)

(continued from previous page)

```
    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)

    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
```

(continues on next page)

(continued from previous page)

```
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.
    """
    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)
```

(continues on next page)

(continued from previous page)

```
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))
    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)
```

(continues on next page)

(continued from previous page)

```
    # 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()
```

5.2.6.2.8 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
```

5.2.6.2.9 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.

5.2.6.2.10 Keywords

Note:

- UserScriptUseCase
- MediumRangeAppUseCase
- NRLOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.6.3 Multi_Tool (MTD): Feature Relative by Lead (with lead groupings)

model_applicaitons/medium_range/ MTD_SeriesAnalysis_fcstGFS _obsGFS_FeatureRelative _SeriesByLead.conf

5.2.6.3.1 Scientific Objective

Demonstrate the capability in the Feature Relative use case but using output from the MET MODE Time Domain (MTD) tool.

5.2.6.3.2 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

5.2.6.3.3 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.

5.2.6.3.4 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

5.2.6.3.5 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]

PROCESS_LIST = MTD, ExtractTiles, SeriesAnalysis

###
# Time Info
###

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)

```
LOOP_ORDER = processes

###
# Field Info
###

MODEL = GFS
OBTYPE = GFS_ANLY

FCST_IS_PROB = False

FCST_VAR1_NAME = PWAT
FCST_VAR1_LEVELS = L0

OBS_VAR1_NAME = PWAT
OBS_VAR1_LEVELS = L0

###
# File I/O
###

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}
```

(continues on next page)

(continued from previous page)

```
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}

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


#######
# MTD Configurations
#######
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}

#######
# ExtractTiles Configurations
#######

EXTRACT_TILES_NLAT = 60
```

(continues on next page)

(continued from previous page)

```
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 Configurations
#######

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}
```

5.2.6.3.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

MTDConfig_wrapped

Note: See the *MTD MET Configuration* (page 172) 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}
```

SeriesAnalysisConfig_wrapped

Note: See the *SeriesAnalysis MET Configuration* (page 207) 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}

```

(continues on next page)

(continued from previous page)

```
//
// 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 = {
   ${METPLUS_OBS_FILE_TYPE}
   ${METPLUS_OBS_CAT_THRESH}
   ${METPLUS_OBS_FIELD}
}

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

(continues on next page)

(continued from previous page)

```
//
// 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 = {
   grid = "";
   poly = "";
}

//
// 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}
```

(continues on next page)

(continued from previous page)

```
//
// 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}
```

5.2.6.3.7 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
```

5.2.6.3.8 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

5.2.6.3.9 Keywords

Note:

- MediumRangeAppUseCase
- TCPairsToolUseCase
- SeriesByLeadUseCase
- MTDToolUseCase
- RegridDataPlaneToolUseCase
- SeriesAnalysisUseCase
- GRIB2FileUseCase
- FeatureRelativeUseCase
- SBUOrgUseCase
- DiagnosticsUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.6.4 Grid-Stat: Standard Verification of Surface Fields

model_applications/medium_range/GridStat_fcstGFS_obsGFS _Sfc_MultiField.conf

5.2.6.4.1 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.

5.2.6.4.2 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 1075) section for more information.

Data Source: GFS

5.2.6.4.3 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.

5.2.6.4.4 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

5.2.6.4.5 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

```
# Grid to Grid Anomoly Example

[config]
# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 2017061300

# End time for METplus run
VALID_END = 2017061306

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 21600

# Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = times

# List of applications to run
PROCESS_LIST = GridStat

# list of variables to compare
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
```

(continues on next page)

(continued from previous page)

```
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

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
```

(continues on next page)

(continued from previous page)

```
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

BOTH_VAR18_NAME = TOZNE
FCST_VAR18_OPTIONS = GRIB_lvl_typ = 200;
BOTH_VAR18_LEVELS = L0

# list of forecast leads to process
LEAD_SEQ = 24

# description of data to be processed
# used in output file path
MODEL = GFS
OBTYPE = ANLYS

# location of grid_stat MET config file
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

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
```

(continues on next page)

(continued from previous page)

```
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

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

# variables to describe format of forecast data
FCST_IS_PROB = false

# variables to describe format of observation data
#  none needed

[dir]

# input and output data directories
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_grid/gfs/fcst
OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_grid/gfs/fcst
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_out/{MODEL}/sfc

[filename_templates]
# format of filenames
# FCST
FCST_GRID_STAT_INPUT_TEMPLATE = pgbf{lead?fmt=%HHH}.gfs.{init?fmt=%Y%m%d%H}

# ANLYS
OBS_GRID_STAT_INPUT_TEMPLATE = pgbf000.gfs.{valid?fmt=%Y%m%d%H}

GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H%M}/grid_stat
```

5.2.6.4.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.6.4.7 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.

5.2.6.4.8 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

5.2.6.4.9 Keywords

Note:

- GridStatToolUseCase
- MediumRangeAppUseCase
- GRIBFileUseCase
- NOAAEMCOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.6.5 Point-Stat: Standard Verification for CONUS Surface

model_applications/medium_range/PointStat_fcstGFS_obsNAM_Sfc _MultiField_PrepBufr.conf

5.2.6.5.1 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.

5.2.6.5.2 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 1088) section for more information.

Data Source: Unknown

5.2.6.5.3 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.

5.2.6.5.4 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

5.2.6.5.5 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]
## Configuration-related settings such as the process list, begin and end times, etc.
PROCESS_LIST = PB2NC, PointStat

## LOOP_ORDER
## Options are: processes, times
## Looping by time- runs all items in the PROCESS_LIST for each
## initialization time and repeats until all times have been evaluated.
## Looping by processes- run each item in the PROCESS_LIST for all
## specified initialization times then repeat for the next item in the
## PROCESS_LIST.
LOOP_ORDER = processes

# Logging levels: DEBUG, INFO, WARN, ERROR (most verbose is DEBUG)
LOG_LEVEL = DEBUG

## MET Configuration files for pb2nc and point_stat
PB2NC_CONFIG_FILE = {PARM_BASE}/met_config/PB2NCConfig_wrapped
POINT_STAT_CONFIG_FILE ={PARM_BASE}/met_config/PointStatConfig_wrapped

PB2NC_SKIP_IF_OUTPUT_EXISTS = True

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20170601
VALID_END = 20170603
VALID_INCREMENT = 86400

LEAD_SEQ = 0


# 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
```

(continues on next page)

(continued from previous page)

```
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

#
# PB2NC
#
# These are appended with PB2NC to differentiate the GRID, POLY, and MESSAGE_TYPE for point_
↪stat.
PB2NC_GRID =
PB2NC_POLY =
PB2NC_STATION_ID =
PB2NC_MESSAGE_TYPE =

# Leave empty to process all
PB2NC_OBS_BUFR_VAR_LIST = PMO, TOB, TDO, UOB, VOB, PWO, TOCC, D_RH

#***********
# ***NOTE***
#***********
# SET TIME_SUMMARY_FLAG to False. There is a bug in met-6.1.
## For defining the time periods for summarization
# False for no time summary, True otherwise
PB2NC_TIME_SUMMARY_FLAG = False
PB2NC_TIME_SUMMARY_BEG = 000000  ;; start time of time summary in HHMMSS format
PB2NC_TIME_SUMMARY_END = 235959  ;; end time of time summary in HHMMSS format
PB2NC_TIME_SUMMARY_VAR_NAMES = PMO,TOB,TDO,UOB,VOB,PWO,TOCC
PB2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80  ;; a list of the␣
↪statistics to summarize

# Model/fcst and obs name, e.g. GFS, NAM, GDAS, etc.
MODEL = gfs
OBTYPE = nam

# Regrid to specified grid.  Indicate NONE if no regridding, or the grid id
# (e.g. G212)
POINT_STAT_REGRID_TO_GRID = G104
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2

# Verification Masking regions
```

(continues on next page)

(continued from previous page)

```
# Indicate which grid and polygon masking region, if applicable
POINT_STAT_GRID = FULL
# List of full path to poly masking files.  NOTE: Only short lists of poly
# files work (those that fit on one line), a long list will result in an
# environment variable that is too long, resulting in an error.  For long
# lists of poly masking files (i.e. all the mask files in the NCEP_mask
# directory), define these in the MET point_stat configuration file.
POINT_STAT_POLY =
POINT_STAT_STATION_ID =

# Message types, if all message types are to be returned, leave this empty,
# otherwise indicate the message types of interest.
POINT_STAT_MESSAGE_TYPE = ONLYSF
# Variables and levels as specified in the field dictionary of the MET
# point_stat configuration file. Specify as FCST_VARn_NAME, FCST_VARn_LEVELS,
# (optional) FCST_VARn_OPTION

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

[dir]
PB2NC_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_obs/prepbufr/nam
PB2NC_OUTPUT_DIR = {OUTPUT_BASE}/nam/conus_sfc

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_obs/gfs
OBS_POINT_STAT_INPUT_DIR = {PB2NC_OUTPUT_DIR}
```

(continues on next page)

(continued from previous page)

```
POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/{OBTYPE}

[filename_templates]
PB2NC_INPUT_TEMPLATE = nam.{da_init?fmt=%Y%m%d}/nam.t{da_init?fmt=%H}z.prepbufr.tm{offset?
↪fmt=%2H}

PB2NC_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/nam.{valid?fmt=%Y%m%d%H}.nc

FCST_POINT_STAT_INPUT_TEMPLATE = pgbf{lead?fmt=%HH}.gfs.{init?fmt=%Y%m%d%H}
OBS_POINT_STAT_INPUT_TEMPLATE = {PB2NC_OUTPUT_TEMPLATE}
```

5.2.6.5.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

PB2NCConfig_wrapped

Note: See the *PB2NC MET Configuration* (page 181) 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 =


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

(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}
```

PointStatConfig_wrapped

Note: See the *PointStat MET Configuration* (page 194) 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
//
```

(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_GRID}
   ${METPLUS_MASK_POLY}
   ${METPLUS_MASK_SID}
   //llpnt =
   ${METPLUS_MASK_LLPNT}
}

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

//
// 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}
```

(continues on next page)

(continued from previous page)

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

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

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

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.6.5.7 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.

5.2.6.5.8 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

5.2.6.5.9 Keywords

Note:

- PB2NCToolUseCase
- MediumRangeAppUseCase
- PointStatToolUseCase
- GRIBFileUseCase
- prepBUFRFileUseCase
- NOAAEMCOrgUseCase
- RegriddinginToolUseCase
- ObsTimeSummaryUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.6.6 Multi_Tool: Feature Relative by Lead (with lead groupings)

model_applicaitons/medium_range/ TCStat_SeriesAnalysis_fcstGFS _obsGFS_FeatureRelative _SeriesByLead.conf

5.2.6.6.1 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.

5.2.6.6.2 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

5.2.6.6.3 External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* netCDF4
```

5.2.6.6.4 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.

5.2.6.6.5 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

5.2.6.6.6 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

```
#
#  CONFIGURATION
#
[config]

# Loop over each process in the process list (set in PROCESS_LIST) for all times in the time␣
↪window of
# interest.
LOOP_ORDER = processes
# Configuration files
TC_PAIRS_CONFIG_FILE = {PARM_BASE}/met_config/TCPairsConfig_wrapped

SERIES_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/SeriesAnalysisConfig_wrapped

PROCESS_LIST = TCPairs, TCStat, ExtractTiles, TCStat(for_series_analysis), SeriesAnalysis

# The init time begin and end times, increment
LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20141214
INIT_END = 20141214

# This is the step-size. Increment in seconds from the begin time to the end
```

(continues on next page)

(continued from previous page)

```
# time
INIT_INCREMENT = 21600 ;; set to every 6 hours=21600 seconds

# Used by extract tiles and series analysis to define the records of
# interest to be retrieved from the grib2 file
#
BOTH_VAR1_NAME = TMP
BOTH_VAR1_LEVELS = Z2

# forecast lead sequence 1 list (0, 6, 12, 18)
LEAD_SEQ_1 = begin_end_incr(0,18,6)
# forecast lead sequence 1 label
LEAD_SEQ_1_LABEL = Day1

# forecast lead sequence 2 list (24, 30, 36, 42)
LEAD_SEQ_2 = begin_end_incr(24,42,6)
# forecast lead sequence 2 label
LEAD_SEQ_2_LABEL = Day2

#######
# TCPairs Configurations
#######

TC_PAIRS_SKIP_LEAD_SEQ = True

# A list of times to include, in format YYYYMMDD_hh
TC_PAIRS_INIT_INCLUDE =

# A list of times to exclude, in format YYYYMMDD_hh
TC_PAIRS_INIT_EXCLUDE =

#
# Specify model valid time window in format YYYYMM[DD[_hh]].  Only tracks
# that fall within the valid time window will
# be used.
#
TC_PAIRS_VALID_BEG =
TC_PAIRS_VALID_END =

#
# Run MET tc_pairs by indicating the top-level directories for the A-deck
# and B-deck files. Set to 'yes' to run using top-level directories, 'no'
# if you want to run tc_pairs on files paired by the wrapper.
TC_PAIRS_READ_ALL_FILES = no
```

(continues on next page)

(continued from previous page)

```
# List of models to be used (white space or comma separated) eg: DSHP, LGEM, HWRF
# If no models are listed, then process all models in the input file(s).
MODEL = GFSO

# List of storm ids of interest (space or comma separated) e.g.: AL112012, AL122012
# If no storm ids are listed, then process all storm ids in the input file(s).
TC_PAIRS_STORM_ID =

# Basins (of origin/region).  Indicate with space or comma-separated list of regions, eg.␣
↪AL: for North Atlantic,
# WP: Western North Pacific, CP: Central North Pacific, SH: Southern Hemisphere, IO: North␣
↪Indian Ocean, LS: Southern
# Hemisphere
TC_PAIRS_BASIN =

# Cyclone, a space or comma-separated list of cyclone numbers. If left empty, all cyclones␣
↪will be used.
TC_PAIRS_CYCLONE =

# Storm name, a space or comma-separated list of storm names to evaluate.  If left empty,␣
↪all storms will be used.
TC_PAIRS_STORM_NAME =

# DLAND file, the full path of the file that contains the gridded representation of the
# minimum distance from land.
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

# overwrite modified track data (non-ATCF to ATCF format)
TC_PAIRS_SKIP_IF_REFORMAT_EXISTS = yes

# overwrite tc_pairs output
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = yes

#######
# TCStat Configurations
#######
# IMPORTANT  Refer to the README_TC for details on setting up analysis
# jobs (located in {MET_INSTALL_DIR}/share/met/config
```

(continues on next page)

(continued from previous page)

```
# Separate each option and value with whitespace, and each job with a whitespace.
# No whitespace within arithmetic expressions or lists of items
# (e.g. -by AMSLP,AMODEL,LEAD -column '(AMAX_WIND-BMAX_WIND)')
# Enclose your arithmetic expressions with '' and separate each job
# by whitespace:
#  -job filter -dump_row /path/to,  -job summary -line_type TCMPR  -column 'ABS(AMAX_WIND-
↪BMAX_WIND)' -out {OUTPUT_BASE}/tc_stat/file.tcst

TC_STAT_JOB_ARGS = -job filter -basin ML -dump_row {TC_STAT_OUTPUT_DIR}/{TC_STAT_DUMP_ROW_
↪TEMPLATE}

# Specify whether only those track points common to both the ADECK and BDECK
# tracks should be written out.  This is only used when explicitly calling
# TC_STAT in the PROCESS_LIST.  This is not used in this use case, so setting
# it to either false or true has no impact.
TC_STAT_MATCH_POINTS = true

# These all map to the options in the default TC-Stat config file, except these
# are pre-pended with TC_STAT to avoid clashing with any other similarly
# named options from other MET tools (eg TC_STAT_AMODEL corresponds to the
# amodel option in the default MET tc-stat config file, whereas AMODEL
# corresponds to the amodel option in the MET tc-pairs config file).

# Stratify by these columns:
TC_STAT_AMODEL =
TC_STAT_BMODEL =
TC_STAT_DESC =
TC_STAT_STORM_ID =
TC_STAT_BASIN =
TC_STAT_CYCLONE =
TC_STAT_STORM_NAME =

# Stratify by init times via a comma-separate list of init times to
# include or exclude.  Time format defined as YYYYMMDD_HH or YYYYMMDD_HHmmss
#TC_STAT_INIT_BEG = 20141213
#TC_STAT_INIT_END = 20141220
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 =
# Stratify by valid times via a comma-separate list of valid times to
# include or exclude.  Time format defined as YYYYMMDD_HH or YYYYMMDD_HHmmss
TC_STAT_VALID_BEG =
TC_STAT_VALID_END =
```

(continues on next page)

(continued from previous page)

```
TC_STAT_VALID_INCLUDE =
TC_STAT_VALID_EXCLUDE =
TC_STAT_VALID_HOUR =
TC_STAT_LEAD_REQ =
TC_STAT_INIT_MASK =
TC_STAT_VALID_MASK =
# Stratify by the valid time and lead time via comma-separated list of
# times in format HH[MMSS]
TC_STAT_VALID_HOUR =
TC_STAT_LEAD =

# Stratify over the watch_warn column in the tcst file.  Setting this to
# 'ALL' will match HUWARN, HUWATCH, TSWARN, TSWATCH
TC_STAT_TRACK_WATCH_WARN =

# Stratify by applying thresholds to numeric data columns.  Specify with
# comma-separated list of column names and thresholds to be applied.
# The length of TC_STAT_COLUMN_THRESH_NAME should be the same as
# TC_STAT_COLUMN_THRESH_VAL.
TC_STAT_COLUMN_THRESH_NAME =
TC_STAT_COLUMN_THRESH_VAL =

# Stratify by a list of comma-separated columns names and values corresponding
# to non-numeric data columns of the values of interest.
TC_STAT_COLUMN_STR_NAME =
TC_STAT_COLUMN_STR_VAL =

# Stratify by applying thresholds to numeric data columns only when lead=0.
# If lead=0 and the value does not meet the threshold, discard the entire
# track.  The length of TC_STAT_INIT_THRESH_NAME must equal the length of
# TC_STAT_INIT_THRESH_VAL.
TC_STAT_INIT_THRESH_NAME =
TC_STAT_INIT_THRESH_VAL =

# Stratify by applying thresholds to numeric data columns only when lead = 0.
# If lead = 0 but the value doesn't meet the threshold, discard the entire
# track.
TC_STAT_INIT_STR_NAME =
TC_STAT_INIT_STR_VAL =

# Excludes any points where distance to land is <=0. When 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.
TC_STAT_WATER_ONLY =
```

(continues on next page)

(continued from previous page)

```
# TRUE or FALSE.  To specify whether only those track points occurring near
# landfall should be retained. Landfall is the last bmodel track point before
# the distance to land switches from water to land.
TC_STAT_LANDFALL =

# Define the landfall retention window, which is defined as the hours offset
# from the time of landfall. Format is in HH[MMSS]. Default TC_STAT_LANDFALL_BEG
# is set to -24, and TC_STAT_LANDFALL_END is set to 00
TC_STAT_LANDFALL_BEG =
TC_STAT_LANDFALL_END =

#######
# ExtractTiles Configurations
#######

# Constants used in creating the tile grid, used by extract tiles
EXTRACT_TILES_NLAT = 60
EXTRACT_TILES_NLON = 60

# Resolution of data in degrees, used by extract tiles
EXTRACT_TILES_DLAT = 0.5
EXTRACT_TILES_DLON = 0.5

# Degrees to subtract from the center lat and lon to
# calculate the lower left lat (lat_ll) and lower
# left lon (lon_ll) for a grid that is 2n X 2m,
# where n = EXTRACT_TILES_LAT_ADJ degrees and m = EXTRACT_TILES_LON_ADJ degrees.
# For this case, where n=15 and m=15, this results
# in a 30 deg X 30 deg grid.  Used by extract tiles
EXTRACT_TILES_LON_ADJ = 15
EXTRACT_TILES_LAT_ADJ = 15

# OVERWRITE OPTIONS
# Skip writing filter files if they already exist.
# Set to yes if you want to skip processing existing files
# Set to no if you want to override existing files
EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS = yes

# 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}
```

(continues on next page)

(continued from previous page)

```
TC_STAT_OUTPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}
TC_STAT_LOOKIN_DIR = {EXTRACT_TILES_OUTPUT_DIR}

[config]

# PLOTTING Relevant to series analysis plots.
# By default, background map is turned off. Set
# to no to turn of plotting of background map.
SERIES_ANALYSIS_BACKGROUND_MAP = no

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False

# set the regrid dictionary item to_grid in the SeriesAnalysis MET config file
SERIES_ANALYSIS_REGRID_TO_GRID = FCST
SERIES_ANALYSIS_REGRID_METHOD = FORCE
#SERIES_ANALYSIS_REGRID_WIDTH =
#SERIES_ANALYSIS_REGRID_VLD_THRESH =
#SERIES_ANALYSIS_REGRID_SHAPE =

## NOTE: "TOTAL" is a REQUIRED cnt statistic used by the series analysis scripts
SERIES_ANALYSIS_STAT_LIST = TOTAL, FBAR, OBAR, ME

SERIES_ANALYSIS_BLOCK_SIZE = 4000

# set to True to add the -paired flag to the SeriesAnalysis command
SERIES_ANALYSIS_IS_PAIRED = True

# If True/yes, run plot_data_plane on output from Series-Analysis to generate
# images for each stat item listed in SERIES_ANALYSIS_STAT_LIST
SERIES_ANALYSIS_GENERATE_PLOTS = yes

# If True/yes, run convert on output from Series-Analysis to generate
# a gif using images in groups of name/level/stat
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}

#  FILENAME TEMPLATES
#
[filename_templates]
# Define the format of the filenames
TC_PAIRS_ADECK_TEMPLATE = {date?fmt=%Y%m}/a{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}
```

(continues on next page)

(continued from previous page)

```
TC_PAIRS_BDECK_TEMPLATE = {date?fmt=%Y%m}/b{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}
TC_PAIRS_OUTPUT_TEMPLATE = {date?fmt=%Y%m}/{basin?fmt=%s}q{date?fmt=%Y%m%d%H}.gfso.{cyclone?
↪fmt=%s}

TC_STAT_DUMP_ROW_TEMPLATE = filter_{init?fmt=%Y%m%d_%H}.tcst

EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}
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_TEMPLATE = {valid?fmt=%Y%m%d}/gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}
↪00_000.grb2

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

FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = {FCST_EXTRACT_TILES_OUTPUT_TEMPLATE}
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = {OBS_EXTRACT_TILES_OUTPUT_TEMPLATE}

SERIES_ANALYSIS_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}

# Template to look for climatology mean input to SeriesAnalysis relative to SERIES_ANALYSIS_
↪CLIMO_MEAN_INPUT_DIR
# Not used in this example
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_TEMPLATE =

# Template to look for climatology standard deviation input to SeriesAnalysis relative to␣
↪SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR
# Not used in this example
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_TEMPLATE =

SERIES_ANALYSIS_OUTPUT_TEMPLATE = {label}/series_F{fcst_beg}_to_F{fcst_end}_{fcst_name}_
↪{fcst_level}.nc


#
#  DIRECTORIES
#
[dir]
TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/track_data
TC_PAIRS_BDECK_INPUT_DIR = {TC_PAIRS_ADECK_INPUT_DIR}
TC_PAIRS_REFORMAT_DIR = {OUTPUT_BASE}/track_data_atcf
TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs
```

(continues on next page)

(continued from previous page)

```
TC_STAT_LOOKIN_DIR = {TC_PAIRS_OUTPUT_DIR}
TC_STAT_OUTPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}

EXTRACT_TILES_TC_STAT_INPUT_DIR = {TC_STAT_OUTPUT_DIR}
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
OBS_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/reduced_model_data
EXTRACT_TILES_OUTPUT_DIR = {OUTPUT_BASE}/extract_tiles

FCST_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
OBS_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
SERIES_ANALYSIS_TC_STAT_INPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}

# directory containing climatology mean input to SeriesAnalysis
# Not used in this example
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_DIR =

# directory containing climatology standard deviation input to SeriesAnalysis
# Not used in this example
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR =

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/series_analysis_lead
```

5.2.6.6.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

TCPairsConfig_wrapped

Note: See the *TCPairs MET Configuration* (page 243) 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 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}
// init_inc =
```

(continues on next page)

(continued from previous page)

```
${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 = TRUE;

//
// 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
```

(continues on next page)

(continued from previous page)

```
//   - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}

//
// 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 256) 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}
```

(continues on next page)

(continued from previous page)

```
//
// 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_INCLUDE}
${METPLUS_INIT_EXCLUDE}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_VALID_INCLUDE}
${METPLUS_VALID_EXCLUDE}

//
// 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 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}
```

(continues on next page)

(continued from previous page)

```
//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}

//
// 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}
```

(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 207) 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}

//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 = {
   grid = "";
   poly = "";
}

//
// 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}
```

(continues on next page)

(continued from previous page)

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

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.6.6.8 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.

5.2.6.6.9 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

5.2.6.6.10 Keywords

Note:

- MediumRangeAppUseCase
- TCPairsToolUseCase
- SeriesByLeadUseCase
- TCStatToolUseCase
- RegridDataPlaneToolUseCase
- MediumRangeAppUseCase
- SeriesAnalysisUseCase
- GRIB2FileUseCase
- FeatureRelativeUseCase
- SBUOrgUseCase
- DiagnosticsUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.6.7 Multi_Tool: Feature Relative by Init

model_applications/medium_range/TCStat_SeriesAnalysis_fcstGFS_obsGFS_FeatureRelative_SeriesByInit.conf

5.2.6.7.1 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.

5.2.6.7.2 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

5.2.6.7.3 External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

```
* netCDF4
```

5.2.6.7.4 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.

5.2.6.7.5 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

5.2.6.7.6 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

```
#
#  CONFIGURATION
#
[config]

# Loop over each process in the process list (set in PROCESS_LIST) for all times in the time␣
↪window of
# interest.
LOOP_ORDER = processes
# Configuration files
TC_PAIRS_CONFIG_FILE = {PARM_BASE}/met_config/TCPairsConfig_wrapped

SERIES_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/SeriesAnalysisConfig_wrapped

PROCESS_LIST = TCPairs, TCStat, ExtractTiles, TCStat(for_series_analysis), SeriesAnalysis

# The init time begin and end times, increment
LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20141214
INIT_END = 20141214

# This is the step-size. Increment in seconds from the begin time to the end
# time
INIT_INCREMENT = 21600 ;; set to every 6 hours=21600 seconds

#######
# TCPairs Configurations
#######

# A list of times to include, in format YYYYMMDD_hh
TC_PAIRS_INIT_INCLUDE =

# A list of times to exclude, in format YYYYMMDD_hh
TC_PAIRS_INIT_EXCLUDE =

# Specify model valid time window in format YYYYMM[DD[_hh]].  Only tracks
# that fall within the valid time window will be used.
TC_PAIRS_VALID_BEG =
TC_PAIRS_VALID_END =
```

(continues on next page)

(continued from previous page)

```
#
# Run MET tc_pairs by indicating the top-level directories for the A-deck
# and B-deck files. Set to 'yes' to run using top-level directories, 'no'
# if you want to run tc_pairs on files paired by the wrapper.
TC_PAIRS_READ_ALL_FILES = no

# List of models to be used (white space or comma separated) eg: DSHP, LGEM, HWRF
# If no models are listed, then process all models in the input file(s).
MODEL = GFSO

# List of storm ids of interest (space or comma separated) e.g.: AL112012, AL122012
# If no storm ids are listed, then process all storm ids in the input file(s).
TC_PAIRS_STORM_ID =

# Basins (of origin/region).  Indicate with space or comma-separated list of regions, eg.␣
↪AL: for North Atlantic,
# WP: Western North Pacific, CP: Central North Pacific, SH: Southern Hemisphere, IO: North␣
↪Indian Ocean, LS: Southern
# Hemisphere
TC_PAIRS_BASIN =

# Cyclone, a space or comma-separated list of cyclone numbers. If left empty, all cyclones␣
↪will be used.
TC_PAIRS_CYCLONE =

# Storm name, a space or comma-separated list of storm names to evaluate.  If left empty,␣
↪all storms will be used.
TC_PAIRS_STORM_NAME =

# DLAND file, the full path of the file that contains the gridded representation of the
# minimum distance from land.
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

# overwrite modified track data (non-ATCF to ATCF format)
TC_PAIRS_SKIP_IF_REFORMAT_EXISTS = yes

# overwrite tc_pairs output
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = yes
```

(continues on next page)

(continued from previous page)

```
#######
# TCStat Configurations
#######
TC_STAT_JOB_ARGS = -job filter -basin ML -dump_row {TC_STAT_OUTPUT_DIR}/{TC_STAT_DUMP_ROW_
↪TEMPLATE}

# Specify whether only those track points common to both the ADECK and BDECK
# tracks should be written out.  This is only used when explicitly calling
# TC_STAT in the PROCESS_LIST.  This is not used in this use case, so setting
# it to either false or true has no impact.
TC_STAT_MATCH_POINTS = true

# These all map to the options in the default TC-Stat config file, except these
# are pre-pended with TC_STAT to avoid clashing with any other similarly
# named options from other MET tools (eg TC_STAT_AMODEL corresponds to the
# amodel option in the default MET tc-stat config file, whereas AMODEL
# corresponds to the amodel option in the MET tc-pairs config file).

# Stratify by these columns:
TC_STAT_AMODEL =
TC_STAT_BMODEL =
TC_STAT_DESC =
TC_STAT_STORM_ID =
TC_STAT_BASIN =
TC_STAT_CYCLONE =
TC_STAT_STORM_NAME =

# Stratify by init times via a comma-separate list of init times to
# include or exclude.  Time format defined as YYYYMMDD_HH or YYYYMMDD_HHmmss
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 =
# Stratify by valid times via a comma-separate list of valid times to
# include or exclude.  Time format defined as YYYYMMDD_HH or YYYYMMDD_HHmmss
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 =
```

(continues on next page)

(continued from previous page)

```
# Stratify by the valid time and lead time via comma-separated list of
# times in format HH[MMSS]
TC_STAT_VALID_HOUR =
TC_STAT_LEAD =

# Stratify over the watch_warn column in the tcst file.  Setting this to
# 'ALL' will match HUWARN, HUWATCH, TSWARN, TSWATCH
TC_STAT_TRACK_WATCH_WARN =

# Stratify by applying thresholds to numeric data columns.  Specify with
# comma-separated list of column names and thresholds to be applied.
# The length of TC_STAT_COLUMN_THRESH_NAME should be the same as
# TC_STAT_COLUMN_THRESH_VAL.
TC_STAT_COLUMN_THRESH_NAME =
TC_STAT_COLUMN_THRESH_VAL =

# Stratify by a list of comma-separated columns names and values corresponding
# to non-numeric data columns of the values of interest.
TC_STAT_COLUMN_STR_NAME =
TC_STAT_COLUMN_STR_VAL =

# Stratify by applying thresholds to numeric data columns only when lead=0.
# If lead=0 and the value does not meet the threshold, discard the entire
# track.  The length of TC_STAT_INIT_THRESH_NAME must equal the length of
# TC_STAT_INIT_THRESH_VAL.
TC_STAT_INIT_THRESH_NAME =
TC_STAT_INIT_THRESH_VAL =

# Stratify by applying thresholds to numeric data columns only when lead = 0.
# If lead = 0 but the value doesn't meet the threshold, discard the entire
# track.
TC_STAT_INIT_STR_NAME =
TC_STAT_INIT_STR_VAL =

# Excludes any points where distance to land is <=0. When 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.
TC_STAT_WATER_ONLY =

# TRUE or FALSE.  To specify whether only those track points occurring near
# landfall should be retained. Landfall is the last bmodel track point before
# the distance to land switches from water to land.
TC_STAT_LANDFALL =

# Define the landfall retention window, which is defined as the hours offset
```

(continues on next page)

(continued from previous page)

```
# from the time of landfall. Format is in HH[MMSS]. Default TC_STAT_LANDFALL_BEG
# is set to -24, and TC_STAT_LANDFALL_END is set to 00
#TC_STAT_LANDFALL_BEG = -24
#TC_STAT_LANDFALL_END = 00
TC_STAT_LANDFALL_BEG =
TC_STAT_LANDFALL_END =

# OVERWRITE OPTIONS
# Skip writing filter files if they already exist.
# Set to yes if you want to skip processing existing files
# Set to no if you want to override existing files
EXTRACT_TILES_SKIP_IF_OUTPUT_EXISTS = yes

# Constants used in creating the tile grid, used by extract tiles
EXTRACT_TILES_NLAT = 60
EXTRACT_TILES_NLON = 60

# Resolution of data in degrees, used by extract tiles
EXTRACT_TILES_DLAT = 0.5
EXTRACT_TILES_DLON = 0.5

# Degrees to subtract from the center lat and lon to
# calculate the lower left lat (lat_ll) and lower
# left lon (lon_ll) for a grid that is 2n X 2m,
# where n = EXTRACT_TILES_LAT_ADJ degrees and m = EXTRACT_TILES_LON_ADJ degrees.
# For this case, where n=15 and m=15, this results
# in a 30 deg X 30 deg grid.  Used by extract tiles
EXTRACT_TILES_LON_ADJ = 15
EXTRACT_TILES_LAT_ADJ = 15

# 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}

[config]

# Used by extract tiles and series analysis to define the records of
# interest to be retrieved from the grib2 file
#
```

(continues on next page)

(continued from previous page)

```
BOTH_VAR1_NAME = TMP
BOTH_VAR1_LEVELS = Z2

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = True

# PLOTTING Relevant to series analysis plots.
# By default, background map is turned off. Set
# to no to turn of plotting of background map.
SERIES_ANALYSIS_BACKGROUND_MAP = no

# set the regrid dictionary item to_grid in the SeriesAnalysis MET config file
SERIES_ANALYSIS_REGRID_TO_GRID = FCST
SERIES_ANALYSIS_REGRID_METHOD = FORCE
#SERIES_ANALYSIS_REGRID_WIDTH =
#SERIES_ANALYSIS_REGRID_VLD_THRESH =
#SERIES_ANALYSIS_REGRID_SHAPE =

## NOTE: "TOTAL" is a REQUIRED cnt statistic used by the series analysis scripts
SERIES_ANALYSIS_STAT_LIST = TOTAL, FBAR, OBAR, ME

SERIES_ANALYSIS_BLOCK_SIZE = 4000

# set to True to add the -paired flag to the SeriesAnalysis command
SERIES_ANALYSIS_IS_PAIRED = True

# If True/yes, run plot_data_plane on output from Series-Analysis to generate
# images for each stat item listed in SERIES_ANALYSIS_STAT_LIST
SERIES_ANALYSIS_GENERATE_PLOTS = yes

# If True/yes, run convert on output from Series-Analysis to generate
# a gif using images in groups of name/level/stat
SERIES_ANALYSIS_GENERATE_ANIMATIONS = no

# Title to use when plotting output from Series-Analysis
# Only used if SERIES_ANALYSIS_GENERATE_PLOTS is True/yes
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}


#
#  FILENAME TEMPLATES
#
[filename_templates]
```

(continues on next page)

(continued from previous page)

```
# Define the format of the filenames
TC_PAIRS_ADECK_TEMPLATE = {date?fmt=%Y%m}/a{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}
TC_PAIRS_BDECK_TEMPLATE = {date?fmt=%Y%m}/b{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}
TC_PAIRS_OUTPUT_TEMPLATE = {date?fmt=%Y%m}/{basin?fmt=%s}q{date?fmt=%Y%m%d%H}.gfso.{cyclone?
↪fmt=%s}

TC_STAT_DUMP_ROW_TEMPLATE = {init?fmt=%Y%m%d_%H}/filter_{init?fmt=%Y%m%d_%H}.tcst

EXTRACT_TILES_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}

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_TEMPLATE = {valid?fmt=%Y%m%d}/gfs_4_{valid?fmt=%Y%m%d}_{valid?fmt=%H}
↪00_000.grb2

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

FCST_SERIES_ANALYSIS_INPUT_TEMPLATE = {FCST_EXTRACT_TILES_OUTPUT_TEMPLATE}
OBS_SERIES_ANALYSIS_INPUT_TEMPLATE = {OBS_EXTRACT_TILES_OUTPUT_TEMPLATE}

SERIES_ANALYSIS_TC_STAT_INPUT_TEMPLATE = {TC_STAT_DUMP_ROW_TEMPLATE}

# Template to look for climatology mean input to SeriesAnalysis relative to SERIES_ANALYSIS_
↪CLIMO_MEAN_INPUT_DIR
# Not used in this example
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_TEMPLATE =

# Template to look for climatology standard deviation input to SeriesAnalysis relative to␣
↪SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR
# Not used in this example
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_TEMPLATE =

SERIES_ANALYSIS_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d_%H}/{storm_id}/series_{fcst_name}_{fcst_
↪level}.nc
#
#  DIRECTORIES
#
[dir]
TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/track_data
TC_PAIRS_BDECK_INPUT_DIR = {TC_PAIRS_ADECK_INPUT_DIR}
```

(continues on next page)

(continued from previous page)

```
TC_PAIRS_REFORMAT_DIR = {OUTPUT_BASE}/track_data_atcf
TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs

TC_STAT_LOOKIN_DIR = {TC_PAIRS_OUTPUT_DIR}
TC_STAT_OUTPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}

EXTRACT_TILES_TC_STAT_INPUT_DIR = {TC_STAT_OUTPUT_DIR}

FCST_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/reduced_model_
↪data
OBS_EXTRACT_TILES_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/reduced_model_data
EXTRACT_TILES_OUTPUT_DIR = {OUTPUT_BASE}/extract_tiles

FCST_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
OBS_SERIES_ANALYSIS_INPUT_DIR = {EXTRACT_TILES_OUTPUT_DIR}
SERIES_ANALYSIS_TC_STAT_INPUT_DIR = {SERIES_ANALYSIS_OUTPUT_DIR}

# directory containing climatology mean input to SeriesAnalysis
# Not used in this example
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_DIR =

# directory containing climatology standard deviation input to SeriesAnalysis
# Not used in this example
SERIES_ANALYSIS_CLIMO_STDEV_INPUT_DIR =

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/series_analysis_init
```

5.2.6.7.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

TCPairsConfig_wrapped

Note: See the *TCPairs MET Configuration* (page 243) 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 = TRUE;

//
// 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:
```

(continues on next page)

(continued from previous page)

```
//    - Input watch/warning filename
//    - Watch/warning time offset in seconds
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}

//
// 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 256) 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}
```

(continues on next page)

(continued from previous page)

```
${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_INCLUDE}
${METPLUS_INIT_EXCLUDE}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_VALID_END}
${METPLUS_VALID_INCLUDE}
${METPLUS_VALID_EXCLUDE}

//
// 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 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 =
```

(continues on next page)

(continued from previous page)

```
${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}

//
// 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.
//
```

(continues on next page)

(continued from previous page)

```
${METPLUS_MATCH_POINTS}

//
// 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 207) 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}

//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 = {
   grid = "";
   poly = "";
}

//
// 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}
```

(continues on next page)

(continued from previous page)

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

//hss_ec_value =
${METPLUS_HSS_EC_VALUE}
rank_corr_flag = FALSE;

tmp_dir = "${MET_TMP_DIR}";

//version        = "V10.0";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.6.7.8 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.

5.2.6.7.9 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

5.2.6.7.10 Keywords

Note:

- TCStatToolUseCase
- SeriesByInitUseCase
- RegridDataPlaneToolUseCase
- MediumRangeAppUseCase
- SeriesAnalysisUseCase
- GRIB2FileUseCase
- TCPairsToolUseCase
- FeatureRelativeUseCase
- SBUOrgUseCase
- DiagnosticsUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.6.8 Grid-Stat: Compute Anomaly Correlation using Climatology

model_applications/medium_range/GridStat_fcstGFS_obsGFS _climoNCEP_MultiField.conf

5.2.6.8.1 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.

5.2.6.8.2 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 1148) section for more information.
Data Source: Unknown

5.2.6.8.3 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.

5.2.6.8.4 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

5.2.6.8.5 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

```
# Grid to Grid Anomoly Example

[config]
# List of applications to run
PROCESS_LIST = GridStat, StatAnalysis

# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 2017061300

# End time for METplus run
VALID_END = 2017061306

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 21600

# list of forecast leads to process
LEAD_SEQ = 24, 48

# Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = times

# list of variables to compare
BOTH_VAR1_NAME = TMP
BOTH_VAR1_LEVELS = P850, P500, P250
```

(continues on next page)

(continued from previous page)

```
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

# description of data to be processed
# used in output file path
MODEL = GFS
OBTYPE = ANLYS

# location of grid_stat MET config file
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

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

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
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_CLIMO_MEAN_DAY_INTERVAL = 1

GRID_STAT_MET_CONFIG_OVERRIDES = climo_mean = fcst;

# variables to describe format of forecast data
FCST_IS_PROB = false

# variables to describe format of observation data
#  none needed

# StatAnalysis configuration
MODEL1 = GFS
MODEL1_OBTYPE = ANLYS

# configuration file to use with StatAnalysis
STAT_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/STATAnalysisConfig_wrapped

# stat_analysis job info
STAT_ANALYSIS_JOB_NAME = filter

# 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 = -dump_row [dump_row_file]

# Optional variables for further filtering
# can be blank, single, or multiple values
# if more than one use comma separated list
#
# (FCST)(OBS)_(VALID)(INIT)_HOUR_LIST: HH format (ex. 00, 06, 12)
# (FCST)(OBS)_LEAD_LIST: HH[H][MMSS] format (ex. 00, 06, 120)
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 =
FCST_LEVEL_LIST =
OBS_LEVEL_LIST =
```

(continues on next page)

(continued from previous page)

```
VX_MASK_LIST =
INTERP_MTHD_LIST =
INTERP_PNTS_LIST =
FCST_THRESH_LIST =
OBS_THRESH_LIST =
COV_THRESH_LIST =
ALPHA_LIST =
LINE_TYPE_LIST =
# how to treat items listed in above _LIST variables
# GROUP_LIST_ITEMS: items listed in a given _LIST variable
#                   will be grouped together
# LOOP_LIST_ITEMS:  items listed in a give _LIST variable
#                   will be looped over
# if not listed METplus will treat the list as a group
GROUP_LIST_ITEMS = FCST_INIT_HOUR_LIST
LOOP_LIST_ITEMS = FCST_VALID_HOUR_LIST, MODEL_LIST


[dir]

# directory containing climatology data
GRID_STAT_CLIMO_MEAN_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_grid/
↪nwprod/fix

# input and output data directories
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_grid/gfs/fcst
OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_grid/gfs/obs
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/met_out/{MODEL}/anom

# directory to look for input for StatAnalysis
MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {OUTPUT_BASE}/met_out/{MODEL1}/anom/*/grid_stat

# Output data directory
STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/gather_by_date/stat_analysis/grid2grid/anom

[filename_templates]
# format of filenames

# Climatology mean
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE = cmean_1d.1959{valid?fmt=%m%d}

# GFS
FCST_GRID_STAT_INPUT_TEMPLATE = pgbf{lead?fmt=%.3H}.gfs.{init?fmt=%Y%m%d%H}

# ANLYS
```

(continues on next page)

(continued from previous page)

```
OBS_GRID_STAT_INPUT_TEMPLATE = pgbanl.gfs.{valid?fmt=%Y%m%d%H}

GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d%H%M}/grid_stat

# Optional settings to create templated directory and file name information
# to save files as stat_analysis output as, this is appended to STAT_ANALYSIS_OUTPUT_DIR
# if no template is provided a default filename set in the code will be used
# Use:
# string templates can be set for all the lists being looped over, just
# use and a lower case version of the list, ex. {fcst_valid_hour?fmt=%H}
# or {fcst_var?fmt=%s}
# For looping over models:
# can set MODELn_STAT_ANALYSIS_[DUMP_ROW/OUT_STAT]_TEMPLATE for individual models
# or STAT_ANALYSIS_[DUMP_ROW/OUT_STAT] with {model?fmt=%s}
MODEL1_STAT_ANALYSIS_DUMP_ROW_TEMPLATE = {fcst_valid_hour?fmt=%H}Z/{MODEL1}/{MODEL1}_{valid?
↪fmt=%Y%m%d}.stat
```

5.2.6.8.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

GridStatConfig_wrapped

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//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 135) 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}
${METPLUS_OBS_LEVEL}
```

(continues on next page)

(continued from previous page)

```
${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}
```

5.2.6.8.7 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.

5.2.6.8.8 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

5.2.6.8.9 Keywords

Note:

- GridStatToolUseCase
- MediumRangeAppUseCase
- StatAnalysisToolUseCase
- GRIBFileUseCase
- NOAAEMCOrgUseCase
- RegriddinginTool

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.6.9 Point-Stat: Standard Verification of Global Upper Air

model_applications/medium_range/PointStat_fcstGFS_obsGDAS _UpperAir_MultiField _PrepBufr.conf

5.2.6.9.1 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.

5.2.6.9.2 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 1161) section for more information.

5.2.6.9.3 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.

5.2.6.9.4 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

5.2.6.9.5 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]
## Configuration-related settings such as the process list, begin and end times, etc.
PROCESS_LIST = PB2NC, PointStat

## LOOP_ORDER
## Options are: processes, times
## Looping by time- runs all items in the PROCESS_LIST for each
## initialization time and repeats until all times have been evaluated.
## Looping by processes- run each item in the PROCESS_LIST for all
## specified initialization times then repeat for the next item in the
## PROCESS_LIST.
LOOP_ORDER = times

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d
VALID_BEG = 20170601
VALID_END = 20170603
VALID_INCREMENT = 86400

LEAD_SEQ = 0

# For both pb2nc and point_stat, the obs_window dictionary:
OBS_WINDOW_BEGIN = -2700
OBS_WINDOW_END = 2700

# Logging levels: DEBUG, INFO, WARN, ERROR (most verbose is DEBUG)
LOG_LEVEL = DEBUG

PB2NC_SKIP_IF_OUTPUT_EXISTS = True

## MET Configuration files for pb2nc and point_stat
PB2NC_CONFIG_FILE = {PARM_BASE}/met_config/PB2NCConfig_wrapped
POINT_STAT_CONFIG_FILE ={PARM_BASE}/met_config/PointStatConfig_wrapped

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
```

(continues on next page)

(continued from previous page)

```
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

#
# PB2NC
#
# These are appended with PB2NC to differentiate the GRID, POLY, and MESSAGE_TYPE for point_
↪stat.
PB2NC_GRID =
PB2NC_POLY =
PB2NC_STATION_ID =
PB2NC_MESSAGE_TYPE =

# Leave empty to process all
PB2NC_OBS_BUFR_VAR_LIST = QOB, TOB, ZOB, UOB, VOB, D_RH

#***********
# ***NOTE***
#***********
# SET TIME_SUMMARY_FLAG to False. There is a bug in met-6.1.
## For defining the time periods for summarization
# False for no time summary, True otherwise
PB2NC_TIME_SUMMARY_FLAG = False
PB2NC_TIME_SUMMARY_BEG = 000000  ;; start time of time summary in HHMMSS format
PB2NC_TIME_SUMMARY_END = 235959  ;; end time of time summary in HHMMSS format
PB2NC_TIME_SUMMARY_VAR_NAMES = PMO,TOB,TDO,UOB,VOB,PWO,TOCC
PB2NC_TIME_SUMMARY_TYPES = min, max, range, mean, stdev, median, p80  ;; a list of the␣
↪statistics to summarize

# Model/fcst and obs name, e.g. GFS, NAM, GDAS, etc.
MODEL = gfs
OBTYPE = gdas

# Regrid to specified grid.  Indicate NONE if no regridding, or the grid id
# (e.g. G212)
POINT_STAT_REGRID_TO_GRID = G003
POINT_STAT_REGRID_METHOD = BILIN
POINT_STAT_REGRID_WIDTH = 2

# Verification Masking regions
# Indicate which grid and polygon masking region, if applicable
```

(continues on next page)

(continued from previous page)

```
POINT_STAT_GRID = FULL
# List of full path to poly masking files.  NOTE: Only short lists of poly
# files work (those that fit on one line), a long list will result in an
# environment variable that is too long, resulting in an error.  For long
# lists of poly masking files (i.e. all the mask files in the NCEP_mask
# directory), define these in the MET point_stat configuration file.
POINT_STAT_POLY =
POINT_STAT_STATION_ID =

# Message types, if all message types are to be returned, leave this empty,
# otherwise indicate the message types of interest.
POINT_STAT_MESSAGE_TYPE = ADPUPA

# Variables and levels as specified in the field dictionary of the MET
# point_stat configuration file. Specify as FCST_VARn_NAME, FCST_VARn_LEVELS,
# (optional) FCST_VARn_OPTION

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

[dir]
PB2NC_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_obs/prepbufr/gdas
PB2NC_OUTPUT_DIR = {OUTPUT_BASE}/gdas/upper_air

FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/grid_to_obs/gfs
OBS_POINT_STAT_INPUT_DIR = {PB2NC_OUTPUT_DIR}

POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/{OBTYPE}
```

(continues on next page)

(continued from previous page)

```
[filename_templates]
## Output file template
PB2NC_INPUT_TEMPLATE = prepbufr.gdas.{valid?fmt=%Y%m%d%H}
PB2NC_OUTPUT_TEMPLATE = prepbufr.gdas.{valid?fmt=%Y%m%d%H}.nc

FCST_POINT_STAT_INPUT_TEMPLATE = pgbf{lead?fmt=%HH}.gfs.{init?fmt=%Y%m%d%H}
OBS_POINT_STAT_INPUT_TEMPLATE = {PB2NC_OUTPUT_TEMPLATE}
```

5.2.6.9.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

PB2NCConfig_wrapped

Note: See the *PB2NC MET Configuration* (page 181) 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";                      },
```

(continues on next page)

(continued from previous page)

```
   { 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;
}

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

//
// Observation types
//
//pb_report_type =
```

(continues on next page)

(continued from previous page)

```
${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 =
${METPLUS_QUALITY_MARK_THRESH}

event_stack_flag    = TOP;
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////////////////////////////////////////////////
//
// 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 194) 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}

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

(continues on next page)

(continued from previous page)

```
//
// 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
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];
```

(continues on next page)

(continued from previous page)

```
//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_GRID}
   ${METPLUS_MASK_POLY}
   ${METPLUS_MASK_SID}
   //llpnt =
   ${METPLUS_MASK_LLPNT}
}

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

//
// 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}

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

//
```

(continues on next page)

(continued from previous page)

```
// Statistical output types
//
//output_flag = {
${METPLUS_OUTPUT_FLAG_DICT}

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

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.6.9.7 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.

5.2.6.9.8 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

5.2.6.9.9 Keywords

Note:

- PB2NCToolUseCase
- PointStatToolUseCase
- MediumRangeAppUseCase
- GRIBFileUseCase
- prepBUFRFileUseCase
- NOAAEMCOrgUseCase
- RegriddinginToolUseCase
- ObsTimeSummaryUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.7 Precipitation

Any fields that can be defined as precipitation, including rain, snow, and other precipitation types

5.2.7.1 Grid-Stat: 6hr QPF in NetCDF format

model_applications/precipitation/GridStat_fcstHREFmean _obsStgIV_Netcdf.conf

5.2.7.1.1 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.

5.2.7.1.2 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

5.2.7.1.3 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.

5.2.7.1.4 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

5.2.7.1.5 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]

PROCESS_LIST = PCPCombine, RegridDataPlane, GridStat

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2017050912
INIT_END=2017050912
INIT_INCREMENT=43200

LEAD_SEQ = 18

LOOP_ORDER = times

FCST_PCP_COMBINE_RUN = True
FCST_PCP_COMBINE_METHOD = ADD

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
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}
```

(continues on next page)

(continued from previous page)

```
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}


FCST_PCP_COMBINE_INPUT_DATATYPE = NETCDF

FCST_IS_PROB = false

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_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

OBS_PCP_COMBINE_INPUT_DATATYPE = NETCDF

REGRID_DATA_PLANE_METHOD = BUDGET
REGRID_DATA_PLANE_WIDTH = 2

REGRID_DATA_PLANE_VERIF_GRID={INPUT_BASE}/model_applications/precipitation/mask/CONUS_
↪HRRRTLE.nc

#USE_EXPLICIT_NAME_AND_LEVEL = True

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 = "(*,*)"
```

(continues on next page)

(continued from previous page)

```
OBS_VAR1_THRESH = gt12.7, gt25.4, gt50.8, gt76.2, gt152.4


GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

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
```

5.2.7.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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 =
${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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.7.1.7 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.

5.2.7.1.8 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

5.2.7.1.9 Keywords

Note:

- GridStatToolUseCase
- PrecipitationAppUseCase
- PCPCombineToolUseCase
- RegridDataPlaneToolUseCase
- NetCDFFileUseCase
- NOAAWPCOrgUseCase
- NOAAHMTOrgUseCase
- NOAAHWTOrgUseCase
- ConvectionAllowingModelsAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.7.2 Ensemble-Stat: WoFS

model_application/precipitation/EnsembleStat_fcstWOFS_obsWOFS.conf

5.2.7.2.1 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.

5.2.7.2.2 Datasets

- Forecast dataset: WoFS Ensemble

5.2.7.2.3 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

5.2.7.2.4 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

5.2.7.2.5 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

```
# pcp_combine  hourly

[config]

LOOP_BY = INIT
#
INIT_TIME_FMT = %Y%m%d%H%M
#
INIT_BEG = 202006151700
#
INIT_END = 202006151700
#
INIT_INCREMENT = 3600
#
LEAD_SEQ = 1

MODEL= WOFS
OBTYPE = MRMS_QPE

# Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
```

(continues on next page)

(continued from previous page)

```
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = times

# List of applications to run
PROCESS_LIST = PcpCombine, EnsembleStat, GridStat

LOG_PCP_COMBINE_VERBOSITY = 3
LOG_ENSEMBLE_STAT_VERBOSITY = 3
LOG_GRID_STAT_VERBOSITY = 3

#### PCP COMBINE ####

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";'

#### ENSEMBLE STAT ####

ENSEMBLE_STAT_N_MEMBERS = 18
ENSEMBLE_STAT_ENS_THRESH = 1.0

ENSEMBLE_STAT_REGRID_TO_GRID = FCST
ENSEMBLE_STAT_REGRID_METHOD = BUDGET
ENSEMBLE_STAT_REGRID_WIDTH = 2
ENSEMBLE_STAT_REGRID_VLD_THRESH = 1.0

ENSEMBLE_STAT_NBRHD_PROB_WIDTH = 1, 3, 5, 7, 9
ENSEMBLE_STAT_NBRHD_PROB_SHAPE = SQUARE
ENSEMBLE_STAT_NBRHD_PROB_VLD_THRESH = 1.0
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_CONFIG_FILE = {PARM_BASE}/met_config/EnsembleStatConfig_wrapped

FCST_ENSEMBLE_STAT_INPUT_GRID_DATATYPE = NETCDF
OBS_ENSEMBLE_STAT_INPUT_GRID_DATATYPE = NETCDF

ENSEMBLE_STAT_NMEP_SMOOTH_VLD_THRESH = 1.0
ENSEMBLE_STAT_NMEP_SMOOTH_SHAPE = SQUARE
ENSEMBLE_STAT_NMEP_SMOOTH_METHOD = NEAREST
ENSEMBLE_STAT_NMEP_SMOOTH_WIDTH = 1
#ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_DX = 3
#ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_RADIUS = 3

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_ENSEMBLE_FLAG_LATLON = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_WEIGHT = FALSE


ENSEMBLE_STAT_OUTPUT_PREFIX = {MODEL}_PCP_{init?fmt=%H%M}_{lead?fmt=%H%M}00L_A1

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
```

(continues on next page)

(continued from previous page)

```
OBS_ENSEMBLE_STAT_VAR1_LEVELS = "(*,*)"
OBS_ENSEMBLE_STAT_VAR1_THRESH = {FCST_ENSEMBLE_STAT_VAR1_THRESH}

#### GRID STAT ####
#
GRID_STAT_REGRID_TO_GRID = FCST

GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

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


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
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}
```

(continues on next page)

(continued from previous page)

```
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
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}
```

(continues on next page)

(continued from previous page)

```
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}
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
```

(continues on next page)

(continued from previous page)

```
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

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

GRID_STAT_OUTPUT_PREFIX = {MODEL}_PCP_{init?fmt=%H%M}_A1
```

(continues on next page)

(continued from previous page)

```
[dir]
FCST_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/WOFS/ensemble
FCST_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/{MODEL}/pcp_combine

FCST_ENSEMBLE_STAT_INPUT_DIR = {FCST_PCP_COMBINE_OUTPUT_DIR}
OBS_ENSEMBLE_STAT_GRID_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/WOFS/OBS
ENSEMBLE_STAT_OUTPUT_DIR = {OUTPUT_BASE}/{MODEL}/ensemble_stat

FCST_GRID_STAT_INPUT_DIR = {ENSEMBLE_STAT_OUTPUT_DIR}
OBS_GRID_STAT_INPUT_DIR = {OBS_ENSEMBLE_STAT_GRID_INPUT_DIR}
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/{MODEL}/grid_stat

[filename_templates]
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_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

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_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_TEMPLATE = {init?fmt=%Y%m%d?shift=-44100}/{init?fmt=%H%M}

FCST_GRID_STAT_INPUT_TEMPLATE = {ENSEMBLE_STAT_OUTPUT_TEMPLATE}/ensemble_stat_{MODEL}_PCP_
↪{init?fmt=%H%M}_{lead?fmt=%H%M}00L_A1_{init?fmt=%Y%m%d}_{valid?fmt=%H%M}00V_ens.nc
OBS_GRID_STAT_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
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d?shift=-44100}/{init?fmt=%H%M}/

GRID_STAT_VERIFICATION_MASK_TEMPLATE = {INPUT_BASE}/model_applications/precipitation/WOFS/
↪domain/WOFS_domain_{init?fmt=%Y%m%d}.nc
```

5.2.7.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

EnsembleStatConfig_wrapped

Note: See the *EnsembleStat MET Configuration* (page 87) 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}

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

//
// May be set separately in each "field" entry
//
${METPLUS_CENSOR_THRESH}
${METPLUS_CENSOR_VAL}
cat_thresh    = [];
nc_var_str    = "";
```

(continues on next page)

(continued from previous page)

```
//
// Ensemble product fields to be processed
//
ens = {

   ${METPLUS_ENS_FILE_TYPE}

   ${METPLUS_ENS_THRESH}
   ${METPLUS_ENS_VLD_THRESH}
   ${METPLUS_ENS_OBS_THRESH}

   ${METPLUS_ENS_FIELD}
}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}

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

//
// Neighborhood ensemble probabilities
//
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
${METPLUS_NMEP_SMOOTH_DICT}

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

//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_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 ];

//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
```

(continues on next page)

(continued from previous page)

```
// 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}



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

(continues on next page)

(continued from previous page)

```
//
${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)

```
//
// Ensemble product output types
//
${METPLUS_ENSEMBLE_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}
```

GridStatConfig_wrapped

Note: See the *GridStat MET Configuration* (page 135) 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 ];
```

(continues on next page)

(continued from previous page)

```
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}

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

//
// Gradient statistics
```

(continues on next page)

(continued from previous page)

```
// 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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.7.2.7 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.

5.2.7.2.8 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

5.2.7.2.9 Keywords

Note:

- EnsembleStatToolUseCase
- PrecipitationAppUseCase
- GRIB2FileUseCase
- EnsembleAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.7.3 Grid-Stat: 6hr PQPF Probability Verification

model_applications/precipitation/GridStat_fcstHRRR-TLE _obsStgIV_GRIB.conf

5.2.7.3.1 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.

5.2.7.3.2 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

5.2.7.3.3 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.

5.2.7.3.4 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

5.2.7.3.5 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]

PROCESS_LIST = PCPCombine, RegridDataPlane, GridStat

LOOP_BY = INIT

INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2016090412
INIT_END=2016090412
INIT_INCREMENT=60

LEAD_SEQ = 6, 7

LOOP_ORDER = times
```

(continues on next page)

(continued from previous page)

```
OBS_PCP_COMBINE_RUN = True
OBS_PCP_COMBINE_METHOD = ADD

OBS_REGRID_DATA_PLANE_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

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}


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_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/GridStat_fcstHRRR-TLE_
↪obsStgIV_GRIB/GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H%M}

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


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
```

(continues on next page)

(continued from previous page)

```
OBS_PCP_COMBINE_INPUT_DATATYPE = GRIB
OBS_PCP_COMBINE_INPUT_ACCUMS = 6, 1


REGRID_DATA_PLANE_METHOD = BUDGET
REGRID_DATA_PLANE_WIDTH = 2

REGRID_DATA_PLANE_VERIF_GRID = {INPUT_BASE}/model_applications/precipitation/mask/CONUS_
↪HRRRTLE.nc


GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

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
```

5.2.7.3.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//grid_weight_flag =
```

(continues on next page)

(continued from previous page)

```
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.7.3.7 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.

5.2.7.3.8 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

5.2.7.3.9 Keywords

Note:

- GridStatToolUseCase
- PrecipitationAppUseCase
- PCPCombineToolUseCase
- RegridDataPlaneToolUseCase
- GRIBFileUseCase
- GRIB2FileUseCase
- NetCDFFileUseCase
- NOAAWPCOrgUseCase
- NOAAHMTOrgUseCase
- NOAAHWTOrgUseCase
- ConvectionAllowingModelsAppUseCase

- ProbabilityVerificationUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.7.4 Point-Stat: Investigating Preciptitation Types

model_application/precipitation/PointStat_fcstMULTI_obsMETAR_PtypeComparisons.conf

5.2.7.4.1 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.

5.2.7.4.2 Datasets

- Forecast dataset: operational GFS, GFSv16, NAM
- Observation dataset: METARs (via NAM prepbufr reanalysis)

5.2.7.4.3 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.

5.2.7.4.4 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

5.2.7.4.5 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]


#The first PointStat call is for the GFS
PROCESS_LIST = PB2NC,PointStat,PointStat(nam_run),PointStat(gfsx_run)
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

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


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%M%S?shift=-1800}
#PB2NC_VALID_END = {valid?fmt=%Y%m%d_%H%M%S?shift=1800}

PB2NC_GRID =
PB2NC_POLY =
PB2NC_STATION_ID =
PB2NC_MESSAGE_TYPE = ADPSFC

PB2NC_PB_REPORT_TYPE =

#PB2NC_LEVEL_RANGE_BEG =
```

(continues on next page)

(continued from previous page)

```
#PB2NC_LEVEL_RANGE_END =

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

#PB2NC_OBS_BUFR_MAP =
#PB2NC_OBS_PREPBUFR_MAP =


###
# 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
POINT_STAT_OUTPUT_PREFIX = gfs

MODEL = gfs

FCST_VAR1_NAME = CRAIN
FCST_VAR1_LEVELS = L0
```

(continues on next page)

(continued from previous page)

```
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

OBS_VAR4_NAME = PRWE
OBS_VAR4_LEVELS = Z0
OBS_VAR4_THRESH = >=174&&<=176

OBS_POINT_STAT_WINDOW_BEGIN = -1800
OBS_POINT_STAT_WINDOW_END = 1800

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
```

(continues on next page)

(continued from previous page)

```
[nam_run]
MODEL = nam
FCST_POINT_STAT_INPUT_TEMPLATE=nam.t12z.awip32{lead?fmt=%2H}.tm00.grib2
POINT_STAT_OUTPUT_PREFIX = nam
```

5.2.7.4.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

PB2NCConfig_wrapped

Note: See the *PB2NC MET Configuration* (page 181) 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";                      }
];
```

(continues on next page)

(continued from previous page)

```
//
// 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;
}

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

//
// Observation types
//
//pb_report_type =
${METPLUS_PB_REPORT_TYPE}

in_report_type  = [];
```

(continues on next page)

(continued from previous page)

```
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 =
${METPLUS_QUALITY_MARK_THRESH}

event_stack_flag    = TOP;

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

(continues on next page)

(continued from previous page)

```
// 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 194) 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
```

(continues on next page)

(continued from previous page)

```
//
// 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
//
// message_type =
${METPLUS_MESSAGE_TYPE}
sid_exc        = [];

//obs_quality_inc =
${METPLUS_OBS_QUALITY_INC}
```

(continues on next page)

(continued from previous page)

```
//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}

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

//
// Verification masking regions
//
```

(continues on next page)

(continued from previous page)

```
mask = {
   ${METPLUS_MASK_GRID}
   ${METPLUS_MASK_POLY}
   ${METPLUS_MASK_SID}
   //llpnt =
   ${METPLUS_MASK_LLPNT}
}

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

//
// 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 = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_OUTPUT_FLAG_DICT}

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

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.7.4.7 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 PointStat_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.

5.2.7.4.8 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).

5.2.7.4.9 Keywords

Note:

- PointStatToolUseCase
- PB2NCToolUseCase
- PrecipitationAppUseCase
- GRIB2FileUseCase
- prepBUFRFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.7.5 MTD: Build Revision Series to Evaluate Forecast Consistency

model_applications/precipitation/MTD_fcstHRRR-TLE_FcstOnly _RevisionSeries_GRIB.conf

5.2.7.5.1 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.

5.2.7.5.2 Datasets

- Forecast dataset: HRRR-TLE forecasts in GRIB2

5.2.7.5.3 METplus Components

This use case runs MTD (MODE Time Domain) over multiple forecast leads.

5.2.7.5.4 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

5.2.7.5.5 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

```
# PHPT MTD Single Run Configuration
[config]

# List of applications to run - only MTD for this case
PROCESS_LIST = MTD

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG=2018031300

# End time for METplus run - must match VALID_TIME_FMT
VALID_END=2018031300

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT=86400

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
# begin_end_incr(start, end, step) can be used to create a list of values
# end value is inclusive
# This will create a list containing 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
LEAD_SEQ = begin_end_incr(12, 0, -1)

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
```

(continues on next page)

(continued from previous page)

```
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Name to identify model (forecast) data in output
MODEL = HRRRTLE

# Name to identify observation data in output
OBTYPE = ANALYS

# if true, only process a single data set with MTD
MTD_SINGLE_RUN = True

# data source if running single mode
# FCST or OBS are valid options
MTD_SINGLE_DATA_SRC = FCST

# list of variables to process
FCST_VAR1_NAME = APCP
FCST_VAR1_LEVELS = R001

# location of MODE Time Domain MET config file
# References CONFIG_DIR from the [dir] section
MTD_CONFIG_FILE = {CONFIG_DIR}/MTDConfig_wrapped

# grid to remap data. Value is set as the 'to_grid' variable in the 'regrid' dictionary
# See MET User's Guide for more information
MTD_REGRID_TO_GRID = NONE

# Minimum volume
MTD_MIN_VOLUME = 2000

# convolution radius for forecast data
FCST_MTD_CONV_RADIUS = 15

# convolution threshold for forecast data
FCST_MTD_CONV_THRESH = >=5.0

# set to True if forecast data is probabilistic
FCST_IS_PROB = false

# input data type of forecast data
```

(continues on next page)

(continued from previous page)

```
FCST_MTD_INPUT_DATATYPE = GRIB

# True if probabilistic information is in the GRIB Product Definition Section
FCST_PROB_IN_GRIB_PDS = false

# output prefix to add to output filenames
MTD_OUTPUT_PREFIX =


# End of [config] section and start of [dir] section
[dir]

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/met_config

FCST_MTD_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/PHPT

MTD_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/MTD_fcstHRRR-TLE_FcstOnly_
↪RevisionSeries_GRIB


[filename_templates]
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_TEMPLATE =
```

5.2.7.5.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *MTD MET Configuration* (page 172) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// MODE Time Domain 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
   //

${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
   //
```

(continues on next page)

(continued from previous page)

```
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 = (

      ( -3.0, 0.0 )
      ( -2.0, 0.5 )
      ( -1.0, 0.8 )
```

(continues on next page)

(continued from previous page)

```
      (  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 )

   );

   start_time_delta = (
```

(continues on next page)

(continued from previous page)

```
      ( -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;
   cluster_id   = true;

}
```

(continues on next page)

(continued from previous page)

```
txt_output = {

   attributes_2d   = true;
   attributes_3d   = true;

}


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

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

tmp_dir = "${MET_TMP_DIR}";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.7.5.7 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.

5.2.7.5.8 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

5.2.7.5.9 Keywords

Note:

- MTDToolUseCase
- PrecipitationAppUseCase
- NOAAHMTOrgUseCase
- GRIB2FileUseCase
- NOAAWPCOrgUseCase
- NOAAHMTOrgUseCase
- NOAAHWTOrgUseCase
- ConvectionAllowingModelsAppUseCase
- RevisionSeriesUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.7.6 Grid-Stat: 6hr QPF in GEMPAK format

model_applications/precipitation/GridStat_fcstHREFmean _obsStgIV_Gempak.conf

5.2.7.6.1 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.

5.2.7.6.2 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. . .)

5.2.7.6.3 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

5.2.7.6.4 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.

5.2.7.6.5 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

5.2.7.6.6 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]

PROCESS_LIST = PCPCombine, RegridDataPlane, GridStat

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=2017050912
INIT_END=2017050912
INIT_INCREMENT=43200

LEAD_SEQ = 18

LOOP_ORDER = times


FCST_PCP_COMBINE_RUN = True
FCST_PCP_COMBINE_METHOD = ADD

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
```

(continues on next page)

(continued from previous page)

```
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}


MODEL = HREF_MEAN
OBTYPE = STAGE4

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


FCST_PCP_COMBINE_INPUT_DATATYPE = GEMPAK
FCST_IS_PROB = false
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,*,*)"
```

(continues on next page)

(continued from previous page)

```
REGRID_DATA_PLANE_METHOD = BUDGET
REGRID_DATA_PLANE_WIDTH = 2

REGRID_DATA_PLANE_VERIF_GRID={INPUT_BASE}/model_applications/precipitation/mask/CONUS_
↪HRRRTLE.nc

OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = "({valid?fmt=%Y%m%d_%H%M%S},*,*)"

OBS_PCP_COMBINE_INPUT_DATATYPE = NETCDF


GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

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
```

5.2.7.6.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//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}
```

5.2.7.6.8 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.

5.2.7.6.9 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

5.2.7.6.10 Keywords

Note:

- GridStatToolUseCase
- PrecipitationAppUseCase
- PCPCombineToolUseCase
- RegridDataPlaneToolUseCase
- GEMPAKFileUseCase
- NetCDFFileUseCase
- NOAAWPCOrgUseCase
- NOAAHMTOrgUseCase
- NOAAHWTOrgUseCase
- ConvectionAllowingModelsAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.7.7 Grid-Stat: 24-hour QPF Use Case

model_applications/precipitation/GridStat_fcstGFS _obsCCPA_Grib.conf

5.2.7.7.1 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.

5.2.7.7.2 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. . .)

5.2.7.7.3 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.

5.2.7.7.4 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

5.2.7.7.5 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]

PROCESS_LIST = PCPCombine, GridStat

LOOP_BY = VALID
VALID_TIME_FMT = %Y%m%d%H
VALID_BEG = 2017061300
VALID_END = 2017061300
VALID_INCREMENT = 86400

LEAD_SEQ = 24

LOOP_ORDER = times

FCST_PCP_COMBINE_RUN = True
FCST_PCP_COMBINE_METHOD = SUM

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_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


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
```

(continues on next page)

(continued from previous page)

```
FCST_PCP_COMBINE_INPUT_DATATYPE = GRIB
FCST_IS_PROB = false
FCST_PCP_COMBINE_INPUT_ACCUMS = 6


GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

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_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_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
```

5.2.7.7.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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          = [];
```

(continues on next page)

(continued from previous page)

```
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)

```
//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.7.7.7 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.

5.2.7.7.8 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

5.2.7.7.9 Keywords

Note:

- GridStatToolUseCase
- PrecipitationAppUseCase
- PCPCombineToolUseCase
- GRIBFileUseCase
- NOAAEMCOrgUseCase
- MediumRangeAppUseCase
- MaskingFeatureUseCase
- RegriddingInToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.7.8 MTD: 6hr QPF Use Case

model_applications/precipitation/MTD_fcstHRRR-TLE _obsMRMS.conf

5.2.7.8.1 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.

5.2.7.8.2 Datasets

- Forecast dataset: HRRR-TLE forecasts in GRIB2
- Observation dataset: Multi Radar Multi Sensor (MRMS)

5.2.7.8.3 METplus Components

This use case runs MTD (MODE Time Domain) over multiple forecast leads and compares them to the observational data set.

5.2.7.8.4 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

5.2.7.8.5 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

```
# PHPT vs. QPE Configurations

[config]
# if false, loop by VALID time
```

(continues on next page)

(continued from previous page)

```
LOOP_BY = INIT

# Format of INIT_BEG and INIT_END
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run
INIT_BEG=2017051003

# End time for METplus run
INIT_END=2017051003

# Increment between METplus runs in seconds. Must be >= 60
INIT_INCREMENT=43200

# list of forecast leads to process
LEAD_SEQ = 1,2,3


# Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = times

# List of applications to run
PROCESS_LIST = MTD

# MODE TIME DOMAIN Configuration

# if true, only process a single data set with MTD
MTD_SINGLE_RUN = False

# Data to process in single mode
# FCST and OBS are valid options
MTD_SINGLE_DATA_SRC = OBS

# forecast convolution radius list
FCST_MTD_CONV_RADIUS = 0

# forecast convolution threshold list
FCST_MTD_CONV_THRESH = >=10

# observation convolution radius list
OBS_MTD_CONV_RADIUS = 15
```

(continues on next page)

(continued from previous page)

```
# observation convolution threshold list
OBS_MTD_CONV_THRESH = >=12.7

# list of variables to compare
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

# description of data to be processed
# used in output file path
MODEL = PHPT
OBTYPE = QPE

# location of MODE Time Domain MET config file
MTD_CONFIG_FILE = {CONFIG_DIR}/MTDConfig_wrapped

MTD_REGRID_TO_GRID = OBS

# PHPT Model Options:
FCST_IS_PROB = true

FCST_PROB_IN_GRIB_PDS = true

# QPE Observation Data Parameters
#  none needed

[dir]
# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/met_config

# input and output data directories for each application in PROCESS_LIST
FCST_MTD_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/PHPT
OBS_MTD_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/QPE_Data

MTD_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/MTD_fcstHRRR-TLE_obsMRMS

MTD_OUTPUT_PREFIX = PROB_{MODEL}_{CURRENT_FCST_NAME}_vs_{OBTYPE}_{CURRENT_OBS_NAME}_A
↪{CURRENT_FCST_LEVEL}

[filename_templates]
```

(continues on next page)

(continued from previous page)

```
# format of filenames

# 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

# QPE
OBS_MTD_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/qpe_{valid?fmt=%Y%m%d%H}.nc

MTD_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H%M}
```

5.2.7.8.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *MTD MET Configuration* (page 172) 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}
```

(continues on next page)

(continued from previous page)

```
   //
   // 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}
```

5.2.7.8.7 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.

5.2.7.8.8 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

5.2.7.8.9 Keywords

Note:

- MTDToolUseCase
- PrecipitationAppUseCase
- GRIB2FileUseCase
- NetCDFFileUseCase
- NOAAWPCOrgUseCase
- NOAAHMTOrgUseCase
- NOAAHWTOrgUseCase
- ConvectionAllowingModelsAppUseCase
- ProbabilityVerificationUseCase
- DiagnosticsUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.7.9 Ensemble-Stat: Basic Post-Processing only

model_application/precipitation/EnsembleStat_fcstHRRRE _FcstOnly_NetCDF.conf

5.2.7.9.1 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.

5.2.7.9.2 Datasets

- Forecast dataset: HRRRE 3 member ensemble netcdf 3 hour precipitation accumulation

5.2.7.9.3 METplus Components

This use case runs Ensemble-Stat on HRRRE data from 3 members after running it through pcp_combine to create a 3 hour precipitation accumulation

5.2.7.9.4 METplus Workflow

The following tools are used for each run time: EnsembleStat

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

5.2.7.9.5 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_fcstHRRRE_FcstOnly_NetCDF.conf

```
[config]

## Configuration-related settings such as the process list, begin and end times, etc.
PROCESS_LIST = EnsembleStat

# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG=2019051912

# End time for METplus run - must match INIT_TIME_FMT
INIT_END=2019052000

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT=43200

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 3,6,9,12

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
```

(continues on next page)

(continued from previous page)

```
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# Name to identify model (forecast) data in output
MODEL = HRRRE

OBTYPE = ANALYS

ENSEMBLE_STAT_N_MEMBERS = 3

ENS_VAR1_NAME = APCP_03
ENS_VAR1_LEVELS = "(*,*)"

# The MET ensemble_stat logging level
# 0 quiet to 5 loud, Verbosity setting for MET output, 2 is default.
# This takes precendence over the general MET logging level set in metplus_logging.conf
LOG_ENSEMBLE_STAT_VERBOSITY = 3

# MET Configuration files for EnsembleSTat
ENSEMBLE_STAT_CONFIG_FILE = {CONFIG_DIR}/EnsembleStatConfig_wrapped

ENSEMBLE_STAT_ENS_THRESH = 0.5

ENSEMBLE_STAT_ENS_VLD_THRESH = 1.0

OBS_ENSEMBLE_STAT_WINDOW_BEGIN = -5400
OBS_ENSEMBLE_STAT_WINDOW_END = 5400

ENSEMBLE_STAT_ENSEMBLE_FLAG_LATLON = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY = TRUE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP = FALSE
```

(continues on next page)

(continued from previous page)

```
ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK = FALSE
ENSEMBLE_STAT_ENSEMBLE_FLAG_WEIGHT = FALSE

ENSEMBLE_STAT_OUTPUT_PREFIX = APCP_03

[dir]

CONFIG_DIR={PARM_BASE}/met_config

# input and output directories for ensemble_stat
# Input File Directories, GRID_STAT and POINT_STAT
FCST_ENSEMBLE_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/HRRRE/pcp_
↪combine

# Ensemble stat output directory
ENSEMBLE_STAT_OUTPUT_DIR = {OUTPUT_BASE}/{MODEL}/ensemble

ENSEMBLE_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/precipitation/HRRRE/pcp_combine
ENSEMBLE_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/precipitation/EnsembleSTat_
↪fcstHRRRE_FcstOnly_NetCDF/EnsembleStat

[filename_templates]

# the following template uses begin_end_incr() notation that expands to:
#  hrrre01_{init?fmt=%Y%m%d%H}f{lead?fmt=%HHH}_A03.nc,
#  hrrre02_{init?fmt=%Y%m%d%H}f{lead?fmt=%HHH}_A03.nc,
#  hrrre03_{init?fmt=%Y%m%d%H}f{lead?fmt=%HHH}_A03.nc
FCST_ENSEMBLE_STAT_INPUT_TEMPLATE = hrrrebegin_end_incr(1,3,1,2)_{init?fmt=%Y%m%d%H}f{lead?
↪fmt=%HHH}_A03.nc

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

5.2.7.9.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *EnsembleStat MET Configuration* (page 87) 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}

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

//
// May be set separately in each "field" entry
//
${METPLUS_CENSOR_THRESH}
${METPLUS_CENSOR_VAL}
cat_thresh    = [];
nc_var_str    = "";

//
// Ensemble product fields to be processed
//
```

(continues on next page)

(continued from previous page)

```
ens = {

   ${METPLUS_ENS_FILE_TYPE}

   ${METPLUS_ENS_THRESH}
   ${METPLUS_ENS_VLD_THRESH}
   ${METPLUS_ENS_OBS_THRESH}

   ${METPLUS_ENS_FIELD}
}

//ens_member_ids =
${METPLUS_ENS_MEMBER_IDS}

//control_id =
${METPLUS_CONTROL_ID}

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

//
// Neighborhood ensemble probabilities
//
${METPLUS_NBRHD_PROB_DICT}

//
// NMEP smoothing methods
//
${METPLUS_NMEP_SMOOTH_DICT}

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

//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_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 ];

//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}
```

(continues on next page)

(continued from previous page)

```
   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}

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

//
// Ensemble product output types
//
${METPLUS_ENSEMBLE_FLAG_DICT}
```

(continues on next page)

(continued from previous page)

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

//
// 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}
```

5.2.7.9.7 Running METplus

This use case can be run two ways:

1) Passing in EnsembleStat_fcstHRRRE_FcstOnly_NetCDF.conf then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪EnsembleStat_fcstHRRRE_FcstOnly_NetCDF.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in EnsembleStat_fcstHRRRE_FcstOnly_NetCDF.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/precipitation/
↪EnsembleStat_fcstHRRRE_FcstOnly_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.

5.2.7.9.8 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/EnsembleStat_fcstHRRRE_FcstOnly_NetCDF/EnsembleStat (relative to **OUTPUT_BASE**) The following folder/file combination will be created:

-201905191200

- ensemble_stat_APCP_03_20190519_150000V_ens.nc
- ensemble_stat_APCP_03_20190519_180000V_ens.nc
- ensemble_stat_APCP_03_20190519_210000V_ens.nc
- ensemble_stat_APCP_03_20190520_000000V_ens.nc

-201905200000

- ensemble_stat_APCP_03_20190520_030000V_ens.nc
- ensemble_stat_APCP_03_20190520_060000V_ens.nc
- ensemble_stat_APCP_03_20190520_090000V_ens.nc
- ensemble_stat_APCP_03_20190520_120000V_ens.nc

5.2.7.9.9 Keywords

Note:

- EnsembleStatToolUseCase
- NOAAHWTOrgUseCase
- PrecipitationAppUseCase
- NetCDFFileUseCase
- EnsembleAppUseCase
- ConvectionAllowingModelsAppUseCase
- ProbabilityGenerationAppUseCase
- ListExpansionFeatureUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/precipitation-EnsembleStat_fcstHRRRE_FcstOnly_NetCDF.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.8 Subseasonal to Seasonal

Subseasonal-to-Seasonal model configurations; Lower resolution model configurations (>4km) usually producing forecasts out beyond 14 days and up 1 year

5.2.8.1 TCGen: Genesis Density Function (GDF) and Track Density Function (TDF)

model_applications/s2s/TCGen_fcstGFSO_obsBDECKS_GDF_TDF.conf

5.2.8.1.1 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.

5.2.8.1.2 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

5.2.8.1.3 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
```

5.2.8.1.4 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.

5.2.8.1.5 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.

5.2.8.1.6 METplus Configuration

```
[config]

# Looping by times: steps through each 'task' in the PROCESS_LIST for each
# defined time, and repeats until all times have been evaluated.
LOOP_ORDER = times

# 'Tasks' to be run
PROCESS_LIST = TCGen, UserScript

LOOP_BY = INIT

# The init time
INIT_TIME_FMT = %Y
INIT_BEG = 2016

LOG_TC_GEN_VERBOSITY = 5
LOG_LEVEL=INFO

# optional list of strings to loop over and call the tool multiple times
# value of each item can be referenced in filename templates with {custom?fmt=%s}
TC_GEN_CUSTOM_LOOP_LIST =


# I/O Configurations

# Location of input data directory for track data
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

```

(continues on next page)

(continued from previous page)

```
# Location of input data directory for genesis data
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

# directory to write output files generated by tc_gen
TC_GEN_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/s2s/TCGen_fcstGFSO_obsBDECKS_GDF_TDF/
↪TCGen
TC_GEN_OUTPUT_TEMPLATE = tc_gen_{init?fmt=%Y}


# MET Configurations

TC_GEN_CONFIG_FILE = {PARM_BASE}/met_config/TCGenConfig_wrapped

# The following variables set values in the MET configuration file used by this example
# Leaving these values commented will use the value found in the default MET configuration␣
↪file
# See the MET documentation for this tool for more information on the settings

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

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
```

(continues on next page)

(continued from previous page)

```
# 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 = 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

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 =

# sets METPLUS_LEAD in the wrapped MET config file
LEAD_SEQ =

TC_GEN_VX_MASK =

TC_GEN_BASIN_MASK =

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
```

(continues on next page)

(continued from previous page)

```
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

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

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_FOR_EACH
```

(continues on next page)

(continued from previous page)

```
USER_SCRIPT_INPUT_TEMPLATE = {TC_GEN_OUTPUT_DIR}/tc_gen_{init?fmt=%Y}_pairs.nc

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/model_applications/s2s/TCGen_fcstGFSO_obsBDECKS_
↪GDF_TDF/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}
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}
```

5.2.8.1.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

TCGenConfig_wrapped

Note: See the *TCGen MET Configuration* (page 227) 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.
//
////////////////////////////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
//
// 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
// 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}

//
```

(continues on next page)

(continued from previous page)

```
// 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.
//
// model =
${METPLUS_MODEL}

//
// BEST and operational track storm identifiers
//
```

(continues on next page)

(continued from previous page)

```
// 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}

//
// Forecast and operational lead times in hours
//
// lead =
${METPLUS_LEAD}
```

(continues on next page)

(continued from previous page)

```
//
// 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}

//
// 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}
```

(continues on next page)

(continued from previous page)

```
//
// 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.
//
////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval alpha value
//
// ci_alpha =
```

(continues on next page)

(continued from previous page)

```
${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.
//
// dland_file =
${METPLUS_DLAND_FILE}

//
// Specify the NetCDF file containing a gridded representation of the
// global basins.
```

(continues on next page)

(continued from previous page)

```
//
// 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}
```

5.2.8.1.8 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
```

(continues on next page)

(continued from previous page)

```
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)))
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}")
```

(continues on next page)

(continued from previous page)

```
  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}")

# 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
```

(continues on next page)

(continued from previous page)

```
# 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.
# 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)]
```

(continues on next page)

(continued from previous page)

```
    # 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

  # 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")
```

(continues on next page)

(continued from previous page)

```
    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)
    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
```

(continues on next page)

(continued from previous page)

```
# 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'))
```

5.2.8.1.9 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
```

5.2.8.1.10 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
* tc_gen_2016_cts.txt
* tc_gen_2016.stat
```

5.2.8.1.11 Keywords

Note:

- TCGenToolUseCase
- S2SAppUseCase
- UserScriptUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.8.2 UserScript: Make a Cross Spectra plot

model_applications/ s2s/ UserScript_obsPrecip_obsOnly_CrossSpectraPlot.py

5.2.8.2.1 Scientific Objective

This use case calls the METplotpy space time plot to create a sample cross spectra diagram using sample data created by METcalcpy cross spectra functions

The space time plot and cross spectra calculations were created by Maria Gehne at the Physical Sciences Labratory in NOAA

5.2.8.2.2 Datasets

5.2.8.2.3 METplus Components

This use case runs the UserScript wrapper tool to run a user provided script, in this case, cross_spectra_plot.py.

5.2.8.2.4 METplus Workflow

This use case does not loop but plots the entire time period of data

UserScript

5.2.8.2.5 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_obsPrecip_obsOnly_CrossSpectraPlot.conf

```
[config]

PROCESS_LIST = UserScript

# Note: time looping is not used in this use case
LOOP_BY = REALTIME
VALID_TIME_FMT = %Y
VALID_BEG = 2020

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/model_applications/s2s/UserScript_obsPrecip_
↪obsOnly_CrossSpectraPlot/cross_spectra_plot.py


[user_env_vars]

# Difficulty index specific variables
```

(continues on next page)

(continued from previous page)

```
LOG_FILE = "cross_spectra_plot.log"

LOG_LEVEL = "INFO"

INPUT_FILE_NAMES = {INPUT_BASE}/model_applications/s2s/UserScript_obsPrecip_obsOnly_
↪CrossSpectraPlot/SpaceTimeSpectra_ERAI_P_D200_symm_2spd.nc,{INPUT_BASE}/model_applications/
↪s2s/UserScript_obsPrecip_obsOnly_CrossSpectraPlot/SpaceTimeSpectra_ERAI_TRMM_P_symm_2spd.
↪nc,{INPUT_BASE}/model_applications/s2s/UserScript_obsPrecip_obsOnly_CrossSpectraPlot/
↪SpaceTimeSpectra_ERAI_P_D850_symm_2spd.nc

METPLOTPY_BASE = {METPLUS_BASE}/METplotpy

YAML_CONFIG_NAME = {METPLUS_BASE}/parm/use_cases/model_applications/s2s/UserScript_obsPrecip_
↪obsOnly_CrossSpectraPlot/spectra_plot.yaml

OUTPUT_DIR = {OUTPUT_BASE}/plots/
```

5.2.8.2.6 MET Configuration

There are no MET tools used in this use case.

5.2.8.2.7 Python Embedding

There is no python embedding in this use case

5.2.8.2.8 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 xarray as xr
```

(continues on next page)

(continued from previous page)

```
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("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)

# 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 = 2

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(",")
#filenames = ['ERAI_TRMM_P_symm_'+str(spd)+'spd',
#             'ERAI_P_D850_symm_'+str(spd)+'spd',
#             'ERAI_P_D200_symm_'+str(spd)+'spd']
vars1 = ['ERAI P', 'ERAI P', 'ERAI P']
vars2 = ['TRMM', 'ERAI D850', 'ERAI D200']
nplot = len(vars1)

for pp in np.arange(0, nplot, 1):

    # read data from file
    var1 = vars1[pp]
    var2 = vars2[pp]
    print("Filename ",filenames[pp])
```

(continues on next page)

(continued from previous page)

```
    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)
    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:
        Coh2 = np.empty([nplot, len(freq[ifreq]), len(wnum[iwave])])
        Phs1 = np.empty([nplot, len(freq[ifreq]), len(wnum[iwave])])
        Phs2 = np.empty([nplot, len(freq[ifreq]), len(wnum[iwave])])
        Pow1 = np.empty([nplot, len(freq[ifreq]), len(wnum[iwave])])
        Pow2 = np.empty([nplot, len(freq[ifreq]), len(wnum[iwave])])
        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):
```

(continues on next page)

(continued from previous page)

```
    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, nplot, 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):
    print(f"ERROR: Could not create output file: {expected_file}")
    sys.exit(1)
```

5.2.8.2.9 Running METplus

This use case can be run two ways:

1) Passing in UserScript_obsPrecip_obsOnly_CrossSpectraPlot.conf, then a user-specific system configuration file:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s/UserScript_
↪obsPrecip_obsOnly_CrossSpectraPlot.conf -c /path/to/user_system.conf
```

2) Modifying the configurations in parm/metplus_config, then passing in UserScript_obsPrecip_obsOnly_CrossSpectraPlot.conf:

```
run_metplus.py -c /path/to/METplus/parm/use_cases/model_applications/s2s/UserScript_
↪obsPrecip_obsOnly_CrossSpectraPlot.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
```

5.2.8.2.10 Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

5.2.8.2.11 Keywords

Note:

- UserScriptUseCase
- S2SAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/UserScript_obsPrecip_obsOnly_CrossSpectraPlot.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.8.3 UserScript: Make a Hovmoeller plot

model_applications/ s2s/ UserScript_obsPrecip_obsOnly_Hovmoeller.py

5.2.8.3.1 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

5.2.8.3.2 Datasets

5.2.8.3.3 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).

5.2.8.3.4 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

5.2.8.3.5 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]

PROCESS_LIST = UserScript

LOOP_BY = REALTIME
VALID_TIME_FMT = %Y
VALID_BEG = 2014

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE

USER_SCRIPT_COMMAND = {PARM_BASE}/use_cases/model_applications/s2s/UserScript_obsPrecip_
↪obsOnly_Hovmoeller/hovmoeller_diagram.py
```

(continues on next page)

(continued from previous page)

```
[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

METPLOTPY_BASE = {METPLUS_BASE}/../METplotpy
OUTPUT_DIR = {OUTPUT_BASE}/plots
```

5.2.8.3.6 MET Configuration

There are no MET tools used in this use case.

5.2.8.3.7 Python Embedding

There is no python embedding in this use case

5.2.8.3.8 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
```

5.2.8.3.9 Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

5.2.8.3.10 Keywords

Note:

- UserScriptUseCase
- S2SAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.8.4 UserScript: Make zonal and meridonial means

model_applications/ s2s/ UserScript_obsERA_obsOnly_Stratosphere.py

5.2.8.4.1 Scientific Objective

This use case calls functions in METcalcpy to create zonal and meridonial means

5.2.8.4.2 Datasets

SSWC_v1.0_varFull_ERAi_d20130106_s20121107_e20130307_c20160701.nc

5.2.8.4.3 METplus Components

This use case runs the UserScript wrapper tool to run a user provided script, in this case, meridonial.py.

5.2.8.4.4 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

5.2.8.4.5 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]

# time looping - 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
LOOP_BY = REALTIME

# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
```

(continues on next page)

(continued from previous page)

```
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# BLank for this usecase but the parameter still needs to be there
VALID_BEG =

# BLank for this usecase but the parameter still needs to be there
VALID_END =

# BLank for this usecase but the parameter still needs to be there
VALID_INCREMENT =

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ =

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

PROCESS_LIST = UserScript

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE

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"
```

(continues on next page)

(continued from previous page)

```
OUTPUT_DIR = {OUTPUT_BASE}
```

5.2.8.4.6 MET Configuration

There are no MET tools used in this use case.

5.2.8.4.7 Python Embedding

There is no python embedding in this use case

5.2.8.4.8 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
```

5.2.8.4.9 Expected Output

A successful run will output the following both to the screen and to the logfile:

```
INFO: METplus has successfully finished running.
```

5.2.8.4.10 Keywords

Note:

- UserScriptUseCase
- S2SAppUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.8.5 SeriesAnalysis: Standardize ensemble members and calculate probabilistic outputs

model_applications/s2s/SeriesAnalysis_fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool.conf

5.2.8.5.1 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.

5.2.8.5.2 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 1319) section for more information.

Data Source: CPC

5.2.8.5.3 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.

5.2.8.5.4 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.

5.2.8.5.5 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]

PROCESS_LIST = SeriesAnalysis, GenEnsProd, SeriesAnalysis(run_two), GridStat

###
# Time Info
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m
INIT_BEG=198201
INIT_END=201002
INIT_INCREMENT = 1Y

LEAD_SEQ =

LOOP_ORDER = processes

###
# SERIES_ANALYSIS FIELDINFO
###

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE
MODEL = CFSv2
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

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
SERIES_ANALYSIS_OUTPUT_STATS_CNT = TOTAL, FBAR, FSTDEV
SERIES_ANALYSIS_BLOCK_SIZE = 0

###
# File I/O SERIES_ANALYSIS
###

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}
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

###
# File I/O Gen_Ens_Prod
###

GEN_ENS_PROD_INPUT_DIR = {FCST_SERIES_ANALYSIS_INPUT_DIR}

GEN_ENS_PROD_INPUT_TEMPLATE = {FCST_SERIES_ANALYSIS_INPUT_TEMPLATE}

#GEN_ENS_PROD_CTRL_INPUT_DIR =
#GEN_ENS_PROD_CTRL_INPUT_TEMPLATE =

GEN_ENS_PROD_N_MEMBERS = 24

GEN_ENS_PROD_OUTPUT_DIR = {OUTPUT_BASE}/GEP
GEN_ENS_PROD_OUTPUT_TEMPLATE = gen_ens_prod_{init?fmt=%Y%m}_ens.nc

###
# Field Info
###

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
```

(continues on next page)

(continued from previous page)

```
###
# GenEnsProd
###

#LOG_GEN_ENS_PROD_VERBOSITY = 2

MODEL = CFSv2
# 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_CENSOR_THRESH =
#GEN_ENS_PROD_CENSOR_VAL =
GEN_ENS_PROD_NORMALIZE = CLIMO_STD_ANOM
#GEN_ENS_PROD_CAT_THRESH =
#GEN_ENS_PROD_NC_VAR_STR =

GEN_ENS_PROD_ENS_THRESH = 0.3
GEN_ENS_PROD_VLD_THRESH = 0.3

#GEN_ENS_PROD_NBRHD_PROB_WIDTH = 5
#GEN_ENS_PROD_NBRHD_PROB_SHAPE = CIRCLE
#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 = {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_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
```

(continues on next page)

(continued from previous page)

```
#GEN_ENS_PROD_CLIMO_MEAN_HOUR_INTERVAL = 6

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_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_CDF = FALSE

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
#GEN_ENS_PROD_CONTROL_ID =

###
# File I/O Grid_Stat
###

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_OUTPUT_DIR = {OUTPUT_BASE}/GridStat
```

(continues on next page)

(continued from previous page)

```
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m}
GRID_STAT_OUTPUT_PREFIX = {init?fmt=%Y%m}


###
# Field Info GridStat
###

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


###
# Field Info for GridStat
###

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_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

[run_two]
###
# FILE I/O of SeriesAnalysis run_two
###
```

(continues on next page)

(continued from previous page)

```
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_OUTPUT_DIR = {OUTPUT_BASE}/SA_run2
SERIES_ANALYSIS_OUTPUT_TEMPLATE = {INIT_BEG}to{INIT_END}_CFSv2_SA.nc

###
# Field Info for SeriesAnalysis run_two
###
#
#These first entries are empty to override the intial SeriesAnalysis call
#SERIES_ANALYSIS_CUSTOM_LOOP_LIST =

#BOTH_SERIES_ANALYSIS_VAR1_NAME =
#BOTH_SERIES_ANALYSIS_VAR1_LEVELS =

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

###
# SeriesAnalysis General for 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_CLIMO_MEAN_FILE_NAME = {INPUT_BASE}/model_applications/s2s/SeriesAnalysis_
↪fcstCFSv2_obsGHCNCAMS_climoStandardized_MultiStatisticTool/ghcn_cams.1x1.1982-2010.mon.
↪clim.nc
```

(continues on next page)

(continued from previous page)

```
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
SERIES_ANALYSIS_CLIMO_STDEV_FIELD = {name="stddev"; level="(0,*,*)";}
SERIES_ANALYSIS_CLIMO_STDEV_FILE_TYPE = NETCDF_NCCF
SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False
```

5.2.8.5.6 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 = {
   grid = "";
   poly = "";
}

//
// 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}

}
```

(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}

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

//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}
```

(continues on next page)

(continued from previous page)

```
//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}

//
// 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}
```

(continues on next page)

(continued from previous page)

```
   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}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
//
// nc_pairs_flag = {
```

(continues on next page)

(continued from previous page)

```
${METPLUS_NC_PAIRS_FLAG_DICT}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.8.5.7 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
```

5.2.8.5.8 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

5.2.8.5.9 Keywords

Note:

- SeriesAnalysisUseCase
- GenEnsProdUseCase
- GridStatUseCase
- ProbabilityVerificationUseCase

- S2SAppUseCase
- NETCDFFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.8.6 Blocking Calculation: ERA RegridDataPlane, PcpCombine, and Blocking python code

model_applications/ s2s/ UserScript_obsERA_obsOnly_Blocking.py

5.2.8.6.1 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:

5.2.8.6.2 Datasets

- Forecast dataset: None
- Observation dataset: ERA Reanlaysis 500 mb height.

5.2.8.6.3 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

5.2.8.6.4 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

5.2.8.6.5 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.

5.2.8.6.6 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/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.

```
# UserScript wrapper example

[config]
# 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 looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
```

(continues on next page)

(continued from previous page)

```
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 1979120100

# End time for METplus run
VALID_END = 2017022800

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 86400

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0

# Only Process DJF
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:0229"

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/use_cases/model_applications/s2s

# Run the obs data
# A variable set to be used in the pre-processing steps
OBS_RUN = True


# Regrid the observations to 1 degree using regrid_data_plane
[regrid_obs]
# End time for METplus run
VALID_END = 2017022818
```

(continues on next page)

(continued from previous page)

```
# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 21600

# 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
OBS_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Z

# Level of input field to process
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = P500

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = Z500

# Mask to use for regridding
REGRID_DATA_PLANE_VERIF_GRID = latlon 360 90 89 0 -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

# input and output data directories for each application in PROCESS_LIST
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/UserScript_fcstGFS_obsERA_Blocking/ERA/
↪Regrid

# format of filenames
# Input ERA Interim
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


# Perform a sum over the 4 daily times that have been regridded using pcp_combine
# 00, 06, 12, 18 UTC
[daily_mean_obs]
# Start time for METplus run
```

(continues on next page)

(continued from previous page)

```
VALID_BEG = 1979120118

# End time for METplus run
VALID_END = 2017022818

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = {OBS_RUN}

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED
OBS_PCP_COMBINE_METHOD = DERIVE
OBS_PCP_COMBINE_STAT_LIST = MEAN

# field name and level of 1 hr accumulation in forecast files
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 output and set 24 hours as the accumulation
OBS_PCP_COMBINE_OUTPUT_ACCUM = 24
OBS_PCP_COMBINE_DERIVE_LOOKBACK = 24

# Name output variable Z500
OBS_PCP_COMBINE_OUTPUT_NAME = Z500

# Input and output Data directories for each application in PROCESS_LIST
OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking/ERA/Regrid
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking/ERA/Daily

# Input and Output filename templates, 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 = Z500_daily_{valid?fmt=%Y%m%d?shift=-64800}_NH.nc


# Perform a 5 day running mean on the data using pcp_combine
[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"

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = {OBS_RUN}

# method to run pcp_combine on forecast data
```

(continues on next page)

(continued from previous page)

```
# Options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED
OBS_PCP_COMBINE_METHOD = DERIVE
OBS_PCP_COMBINE_STAT_LIST = MEAN

# field name, level and setting time attribute of 1 hr accumulation in forecast files
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

# Set output variable name
OBS_PCP_COMBINE_OUTPUT_NAME = Z500

# input and output data directories for each application in PROCESS_LIST
OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking/ERA/Daily
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking/ERA/Rmean5d

# format of filenames
# Input ERA Interim
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


# 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"

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = {OBS_RUN}

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED
OBS_PCP_COMBINE_METHOD = USER_DEFINED

# User defined pcp_combine command
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="(*,*)";'

# input and output data directories for each application in PROCESS_LIST
```

(continues on next page)

(continued from previous page)

```
OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking/ERA
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking/ERA/Anomaly

# format of filenames
# Input ERA Interim
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

# Maximum number of grid points between IBLs for everything in between to be included as an
↪IBL
OBS_IBL_DIST = 7
```

(continues on next page)

(continued from previous page)

```
# 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/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


# Run the Blocking Analysis Script
[script_blocking]
# 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 pler 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/UserScript_fcstGFS_obsERA_
↪Blocking/ERA/Anomaly/Z500_anomaly_{valid?fmt=%Y%m%d}_NH.nc,{INPUT_BASE}/model_applications/
↪s2s/UserScript_fcstGFS_obsERA_Blocking/ERA/Daily/Z500_daily_{valid?fmt=%Y%m%d}_NH.nc
```

(continues on next page)

(continued from previous page)

```
# 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/UserScript_obsERA_
↪obsOnly_Blocking/Blocking_driver.py
```

5.2.8.6.7 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

5.2.8.6.8 Python Scripts

This use case uses Python scripts to perform the blocking calculation

parm/use_cases/model_applications/s2s/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
```

(continues on next page)

(continued from previous page)

```
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')

    # 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()
```

(continues on next page)

(continued from previous page)

```
    # 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)

    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
```

(continues on next page)

(continued from previous page)

```
        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)


    # 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):
```

(continues on next page)

(continued from previous page)

```
            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_
↪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')
```

(continues on next page)

(continued from previous page)

```
        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')
        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):
```

(continues on next page)

(continued from previous page)

```
            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␣
↪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()
```

5.2.8.6.9 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/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/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
```

5.2.8.6.10 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/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.

5.2.8.6.11 Keywords

Note:

- RegridDataPlaneUseCase
- PCPCombineUseCase
- S2SAppUseCase
- NetCDFFileUseCase
- GRIB2FileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s-OBS_ERA_blocking_frequency.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.8.7 GridStat: Determine dominant ensemble members terciles and calculate categorical outputs

model_applications/s2s/GridStat_fcstCFSv2_obsGHCNCAMS_MultiTercile.conf

5.2.8.7.1 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.

5.2.8.7.2 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 1345) section for more information.

Data Source: CPC

5.2.8.7.3 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.

5.2.8.7.4 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.

5.2.8.7.5 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]

PROCESS_LIST = GridStat

###
# Time Info
###

LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG=1982010100
INIT_END=2010020100
INIT_INCREMENT = 1Y

LEAD_SEQ =

LOOP_ORDER = processes
```

(continues on next page)

(continued from previous page)

```
###
# File I/O
###


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}


###
# 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
###
```

(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_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_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 =



GRID_STAT_OUTPUT_FLAG_MCTC = BOTH
GRID_STAT_OUTPUT_FLAG_MCTS = BOTH

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 =
```

5.2.8.7.6 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}

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

//
// 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}

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

(continues on next page)

(continued from previous page)

```
//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 =
   ${METPLUS_NBRHD_SHAPE}
   // width =
   ${METPLUS_NBRHD_WIDTH}
```

(continues on next page)

(continued from previous page)

```
   // 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}

//
// NetCDF matched pairs output file
// May be set separately in each "obs.field" entry
```

(continues on next page)

(continued from previous page)

```
//
// nc_pairs_flag = {
${METPLUS_NC_PAIRS_FLAG_DICT}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.8.7.7 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/marine_and_cryosphere/
↪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
```

5.2.8.7.8 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

5.2.8.7.9 Keywords

Note:

- GridStatUseCase
- ProbabilityVerificationUseCase
- PythonEmbeddingFileUseCase
- S2SAppUseCase
- NETCDFFileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.8.8 UserScript: Make OMI plot from calculated MJO indices

model_applications/ s2s/ UserScript_obsERA_obsOnly_OMI.py

5.2.8.8.1 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.

5.2.8.8.2 Datasets

- Forecast dataset: None
- Observation dataset: ERA Reanlaysis Outgoing Longwave Radiation.

5.2.8.8.3 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

5.2.8.8.4 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.

5.2.8.8.5 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.

5.2.8.8.6 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/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)

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H
```

(continues on next page)

(continued from previous page)

```
# Start time for METplus run
VALID_BEG = 1979010100

# End time for METplus run
VALID_END = 2012123000

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 86400

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/use_cases/model_applications/s2s

# Run the obs for these cases
OBS_RUN = True
FCST_RUN = False

# Mask to use for regridding
REGRID_DATA_PLANE_VERIF_GRID = latlon 144 17 -20 0 2.5 2.5

# 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

# 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/UserScript_obsERA_obsOnly_OMI/ERA
OBS_OLR_INPUT_TEMPLATE = OLR_{valid?fmt=%Y%m%d}.nc
```

(continues on next page)

(continued from previous page)

```
# Configurations for regrid_data_plane:  Regrid OLR to -20 to 20 latitude
[regrid_obs_olr]
# 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
OBS_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_NAME = olr

# Level of input field to process
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;

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = olr

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/UserScript_obsERA_
↪obsOnly_OMI
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OBS_OLR_INPUT_DIR}

# format of filenames
# Input ERA Interim
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = olr.1x.7920.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {OBS_OLR_INPUT_TEMPLATE}


# 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
```

(continues on next page)

(continued from previous page)

```
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s/UserScript_obsERA_obsOnly_
↪OMI/EOF/eof1/eof{valid?fmt=%j}.txt,{INPUT_BASE}/model_applications/s2s/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
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/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


# 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
```

(continues on next page)

(continued from previous page)

```
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/UserScript_obsERA_
↪obsOnly_OMI/OMI_driver.py
```

5.2.8.8.7 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.

5.2.8.8.8 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/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
```

(continues on next page)

(continued from previous page)

```
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

    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
```

(continues on next page)

(continued from previous page)

```
    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
↪',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():
```

(continues on next page)

(continued from previous page)

```
    # 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)

    #  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)
```

(continues on next page)

(continued from previous page)

```
    # 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()
```

5.2.8.8.9 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/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/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
```

5.2.8.8.10 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/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/UserScript_obsERA_obsOnly_OMI/plots (relative to **OUTPUT_BASE**).

5.2.8.8.11 Keywords

Note:

- S2SAppUseCase
- RegridDataPlaneUseCase
- PCPCombineUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1595) to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s-OMI_phase_diagram.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.8.9 UserScript: Make a Phase Diagram plot from input RMM or OMI

model_applications/ s2s/ UserScript_obsERA_obsOnly_PhaseDiagram.py

5.2.8.9.1 Scientific Objective

To produce a phase diagram using either OLR based MJO Index (OMI) or the Real-time Multivariate MJO index (RMM)

5.2.8.9.2 Datasets

- Forecast dataset: None.
- Observation dataset: ERA Reanlaysis Outgoing Longwave Radiation.

5.2.8.9.3 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

5.2.8.9.4 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.

5.2.8.9.5 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.

5.2.8.9.6 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/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.

```
# OMI UserScript wrapper
[config]
# Steps
PROCESS_LIST = UserScript(obs_time_filelist), UserScript(script_PhaseDiagram)
```

(continues on next page)

(continued from previous page)

```
# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 2012010100

# End time for METplus run
VALID_END = 2012033100

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 86400

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/use_cases/model_applications/s2s

# Run the obs for these cases
OBS_RUN = True
FCST_RUN = False
```

(continues on next page)

(continued from previous page)

```
# 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 = {INPUT_BASE}/model_applications/s2s/UserScript_obsERA_obsOnly_
↪PhaseDiagram/
OBS_PDTIME_OUTPUT_TEMPLATE = time_list_lead{lead?fmt=%HHH}.txt


# 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/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 = {OBS_PDTIME_OUTPUT_DIR}

# 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
PHASE_DIAGRAM_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_obsERA_obsOnly_PhaseDiagram/
↪plots
```

(continues on next page)

(continued from previous page)

```
# Plot Ouptut Name
OBS_PHASE_PLOT_OUTPUT_NAME = RMM_phase_diagram


# 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/UserScript_obsERA_
↪obsOnly_PhaseDiagram/PhaseDiagram_driver.py
```

5.2.8.9.7 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.

5.2.8.9.8 Python Scripts

The phase diagram driver script orchestrates the generation of a phase diagram plot: parm/use_cases/model_applications/s2s/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
"""

import os
import atexit
import numpy as np
import pandas as pd
import datetime
import warnings
```

(continues on next page)

(continued from previous page)

```
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:
        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]:
```

(continues on next page)

(continued from previous page)

```
                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()

    # 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'):
```

(continues on next page)

(continued from previous page)

```
        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]

filelist = open(output_file,'a+')
filelist.write(input_file + '\n')
filelist.close()
```

5.2.8.9.9 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/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/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
```

5.2.8.9.10 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/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/UserScript_obsERA_obsOnly_PhaseDiagram/plots (relative to **OUTPUT_BASE**).

5.2.8.9.11 Keywords

Note:

- S2SAppUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1595) to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/s2s-PhaseDiagram.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.8.10 WeatherRegime Calculation: ERA RegridDataPlane, PcpCombine, and WeatherRegime python code

model_applications/ s2s/ UserScript_obsERA_obsOnly_WeatherRegime.py

5.2.8.10.1 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.

5.2.8.10.2 Datasets

- Forecast dataset: None.
- Observation dataset: ERA Reanlaysis 500 mb height.

5.2.8.10.3 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

5.2.8.10.4 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

5.2.8.10.5 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.

5.2.8.10.6 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/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.

```
# UserScript wrapper for Weather Regime Analysis
[config]
# 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 looping - 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
LOOP_BY = VALID
```

(continues on next page)

(continued from previous page)

```
# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 1979120100

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 2017022800

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 86400

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0

# Only Process DJF
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:0229"

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/use_cases/model_applications/s2s


# Regridding Pre-Processing Step
[regrid_obs]
# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 1979120100

# End time for METplus run - must match VALID_TIME_FMT
```

(continues on next page)

(continued from previous page)

```
VALID_END = 2017022818

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 21600

# REGRID_DATA_PLANE (Pre Processing Step 1), currently turned off
# Run regrid_data_plane on forecast data
OBS_REGRID_DATA_PLANE_RUN = True

# If true, process each field individually and write a file for each
# If false, run once per run time passing in all fields specified
OBS_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Z

# Level of input field to process
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = P500

# Name of output field to create
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

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/UserScript_fcstGFS_
↪obsERA_WeatherRegime/ERA/OrigData
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_WeatherRegime/
↪ERA/Regrid

# format of filenames
# Input and output ERA Interim
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)

```
# Daily Mean Pre-Processing Step
[daily_mean_obs]
# Start time for METplus run
VALID_BEG = 1979120118

# End time for METplus run
VALID_END = 2017022818

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = True

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED
OBS_PCP_COMBINE_METHOD = DERIVE
OBS_PCP_COMBINE_STAT_LIST = MEAN

# field name and level of 1 hr accumulation in forecast files
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

# input and output data directories for each application in PROCESS_LIST
OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_WeatherRegime/ERA/
↪Regrid
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_WeatherRegime/ERA/
↪Daily

# 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 = Z500_daily_{valid?fmt=%Y%m%d?shift=-64800}_NH.nc


# Variables for the Weather Regime code
[user_env_vars]
```

(continues on next page)

(continued from previous page)

```
# 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

# 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/UserScript_fcstGFS_obsERA_WeatherRegime

# Directory to send output plots
WR_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s/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
```

(continues on next page)

(continued from previous page)

```
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/UserScript_fcstGFS_obsERA_WeatherRegime/mpr


# Run the Weather Regime Script
[script_wr]
# Run the user script once
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/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/UserScript_
↪fcstGFS_obsERA_WeatherRegime/WeatherRegime_driver.py
```

5.2.8.10.7 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

5.2.8.10.8 Python Scripts

This use case uses Python scripts to perform the blocking calculation

parm/use_cases/model_applications/s2s/UserScript_obsERA_obsOnly_WeatherRegime/WeatherRegime_driver.py: 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).
↪zfill(2)+'_freq_stat_'+modname)
```

(continues on next page)

(continued from previous page)

```
            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()
```

5.2.8.10.9 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/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/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
```

5.2.8.10.10 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/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.

5.2.8.10.11 Keywords

Note:

- RegridDataPlaneUseCase
- PCPCombineUseCase
- StatAnalysisUseCase

- S2SAppUseCase
- NetCDFFileUseCase
- GRIB2FileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s-OBS_ERA_weather_regime_freq.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.8.11 UserScript: Make RMM plots from calculated MJO indices

model_applications/ s2s/ UserScript_obsERA_obsOnly_RMM.py

5.2.8.11.1 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.

5.2.8.11.2 Datasets

- Forecast dataset: None
- Observation dataset: ERA Reanlaysis Outgoing Longwave Radiation, 850 hPa wind and 200 hPa wind

5.2.8.11.3 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

5.2.8.11.4 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.

5.2.8.11.5 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.

5.2.8.11.6 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/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.

```
# RMM UserScript wrapper
[config]
# 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)
```

(continues on next page)

(continued from previous page)

```
# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 2000010100

# End time for METplus run
VALID_END = 2002123000

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 86400

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/use_cases/model_applications/s2s

# Run the obs for these cases
OBS_RUN = True
FCST_RUN = False

# Mask to use for regridding
```

(continues on next page)

(continued from previous page)

```
REGRID_DATA_PLANE_VERIF_GRID = latlon 144 13 -15 0 2.5 2.5

# 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


# 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

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
# Set to one year, since we want a mean daily across all years
# Using 2012 because leap day will be included
VALID_BEG = 2012010100

# End time for METplus run
VALID_END = 2012123100

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 86400

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = {OBS_RUN}

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED
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/UserScript_obsERA_obsOnly_
↪RMM/ERA/daily_mean
```

(continues on next page)

(continued from previous page)

```
OBS_PCP_COMBINE_INPUT_TEMPLATE = ERA_wind_daily_mean_*{valid?fmt=%m%d}.nc

OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_obsERA_obsOnly_RMM/ERA/mean_daily_
↪annual_cycle
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = ERA_wind_daily_annual_{valid?fmt=%m%d}.nc


# 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

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
# Set to one year, since we want a mean daily across all years
# Using 2012 because leap day will be included
VALID_BEG = 2012010100

# End time for METplus run
VALID_END = 2012123100

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 86400

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = {OBS_RUN}

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED
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="(*,*)";'

OBS_PCP_COMBINE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/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/UserScript_obsERA_obsOnly_RMM/ERA/mean_daily_
↪annual_cycle
```

(continues on next page)

(continued from previous page)

```
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = ERA_OLR_daily_annual_{valid?fmt=%m%d}.nc


# Configurations for creating U200 and U850 daily mean obs
[daily_mean_obs_wind]
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 1979010100

# End time for METplus run
VALID_END = 2002123100

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 86400

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = {OBS_RUN}

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED
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/UserScript_obsERA_obsOnly_RMM/ERA/daily_mean
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = ERA_wind_daily_mean_{valid?fmt=%Y%m%d}.nc


# Configurations for creating mean daily annual cycle obs OLR
[daily_mean_obs_olr]
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
```

(continues on next page)

(continued from previous page)

```
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 1979010100

# End time for METplus run
VALID_END = 2002123100

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 86400

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = {OBS_RUN}

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED
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/UserScript_obsERA_obsOnly_RMM/ERA/daily_mean
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = ERA_OLR_daily_mean_{valid?fmt=%Y%m%d}.nc


# 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/UserScript_obsERA_obsOnly_
↪RMM/ERA/mean_daily_annual_cycle/ERA_OLR_daily_annual_{valid?fmt=%m%d}.nc,{INPUT_BASE}/
↪model_applications/s2s/UserScript_obsERA_obsOnly_RMM/ERA/mean_daily_annual_cycle/ERA_wind_
↪daily_annual_{valid?fmt=%m%d}.nc
```

(continues on next page)

(continued from previous page)

```
# 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


# 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/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/UserScript_obsERA_
↪obsOnly_RMM/compute_harmonic_anomalies.py 'METPLUS_FILELIST_INPUT_MEAN_DAILY_ANNUAL_
↪INFILES_OLR' 'olr' 'olr_NA_mean' '{OUTPUT_BASE}/s2s/UserScript_obsERA_obsOnly_RMM/ERA/
↪Anomaly' 'ERA_OLR_anom'


# 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/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
```

(continues on next page)

(continued from previous page)

```
# 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/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/UserScript_obsERA_obsOnly_RMM/
↪ERA/Anomaly' 'ERA_U850_anom'


# 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/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/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/UserScript_obsERA_obsOnly_RMM/
↪ERA/Anomaly' 'ERA_U200_anom'


# Configurations for regrid_data_plane: Regrid OLR to -15 to 15 latitude
[regrid_obs_olr]
# 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
```

(continues on next page)

(continued from previous page)

```
# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_NAME = olr_anom

# Level of input field to process
OBS_REGRID_DATA_PLANE_VAR1_LEVELS = "(*,*)"

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = OLR_anom

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_obsERA_obsOnly_RMM/ERA/Anomaly
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_obsERA_obsOnly_RMM/ERA/Regrid

# format of filenames
# Input ERA Interim
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


# Configurations for regrid_data_plane: Regrid u850 to -15 to 15 latitude
[regrid_obs_u850]
# 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 = U_P850_mean_anom

# Level of input field to process
OBS_REGRID_DATA_PLANE_VAR1_LEVELS = "(*,*)"

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = U_P850_anom

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_obsERA_obsOnly_RMM/ERA/Anomaly
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_obsERA_obsOnly_RMM/ERA/Regrid

# format of filenames
# Input ERA Interim
```

(continues on next page)

(continued from previous page)

```
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


# Configurations for regrid_data_plane: Regrid u200 to -15 to 15 latitude
[regrid_obs_u200]
# 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 = U_P200_mean_anom

# Level of input field to process
OBS_REGRID_DATA_PLANE_VAR1_LEVELS = "(*,*)"

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = U_P200_anom

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_obsERA_obsOnly_RMM/ERA/Anomaly
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_obsERA_obsOnly_RMM/ERA/Regrid

# format of filenames
# Input ERA Interim
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
[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
```

(continues on next page)

(continued from previous page)

```
# 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/UserScript_obsERA_obsOnly_RMM/
↪EOF/rmm_olr_eofs.txt
U850_EOF_INPUT_TEXTFILE = {INPUT_BASE}/model_applications/s2s/UserScript_obsERA_obsOnly_RMM/
↪EOF/rmm_u850_eofs.txt
U200_EOF_INPUT_TEXTFILE = {INPUT_BASE}/model_applications/s2s/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/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

# 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


# Configurations for UserScript: Run the RMM Analysis driver
```

(continues on next page)

(continued from previous page)

```
[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/UserScript_obsERA_obsOnly_RMM/ERA/Regrid/ERA_
↪OLR_{valid?fmt=%Y%m%d}.nc,{OUTPUT_BASE}/s2s/UserScript_obsERA_obsOnly_RMM/ERA/Regrid/ERA_
↪U850_{valid?fmt=%Y%m%d}.nc,{OUTPUT_BASE}/s2s/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/UserScript_obsERA_
↪obsOnly_RMM/RMM_driver.py
```

5.2.8.11.7 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.

5.2.8.11.8 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/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/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
```

(continues on next page)

(continued from previous page)

```
"""

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␣
↪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
```

(continues on next page)

(continued from previous page)

```
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:]

    # 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)
```

(continues on next page)

(continued from previous page)

```
    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')
    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)
```

(continues on next page)

(continued from previous page)

```
    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)

    # 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
```

(continues on next page)

(continued from previous page)

```
    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
    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')
```

(continues on next page)

(continued from previous page)

```
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'

# 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 = []
```

(continues on next page)

(continued from previous page)

```
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)
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)
```

(continues on next page)

(continued from previous page)

```
    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)
    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'))
```

5.2.8.11.9 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/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/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
```

5.2.8.11.10 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/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/UserScript_obsERA_obsOnly_RMM/plots (relative to **OUTPUT_BASE**).

5.2.8.11.11 Keywords

Note:

- S2SAppUseCase
- NetCDFFileUseCase
- RegridDataPlaneUseCase
- PCPCombineUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1595) to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/s2s-RMM_time_series.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.8.12 WeatherRegime Calculation: GFS and ERA RegridDataPlane, PcpCombine, and WeatherRegime python code

model_applications/ s2s/ UserScript_fcstGFS_obsERA_WeatherRegime.py

5.2.8.12.1 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.

5.2.8.12.2 Datasets

- Forecast dataset: GFS Forecast 500 mb height.
- Observation dataset: ERA Reanlaysis 500 mb height.

5.2.8.12.3 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

5.2.8.12.4 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

5.2.8.12.5 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.

5.2.8.12.6 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/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.

```
# UserScript wrapper for Weather Regime Analysis
[config]
# 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 looping - 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)

```
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
#VALID_BEG = 1979120100
VALID_BEG = 2000120100

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 2017022800

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 86400

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0

# Only Process DJF
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:1201,0229"

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/use_cases/model_applications/s2s


# Regridding Pre-Processing Step
[regrid_obs]
# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 2000120200
```

(continues on next page)

(continued from previous page)

```
# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 2017022818

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 21600

# REGRID_DATA_PLANE (Pre Processing Step 1), currently turned off
# Run regrid_data_plane on forecast data
OBS_REGRID_DATA_PLANE_RUN = True

# If true, process each field individually and write a file for each
# If false, run once per run time passing in all fields specified
OBS_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Z

# Level of input field to process
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = P500

# Name of output field to create
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

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/UserScript_fcstGFS_
↪obsERA_WeatherRegime/ERA/OrigData
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_WeatherRegime/
↪ERA/Regrid

# format of filenames
# Input and output ERA Interim
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = {valid?fmt=%Y%m}/ei.oper.an.pl.regn128sc.{valid?fmt=%Y
↪%m%d%H}
```

(continues on next page)

(continued from previous page)

```
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {valid?fmt=%Y%m%d}/Z500_6hourly_{init?fmt=%Y%m%d%H}_
↪NH.nc


# Daily Mean Pre-Processing Step
[daily_mean_obs]
# Start time for METplus run
VALID_BEG = 2000120218

# End time for METplus run
VALID_END = 2017022818

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = True

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED
OBS_PCP_COMBINE_METHOD = DERIVE
OBS_PCP_COMBINE_STAT_LIST = MEAN

# field name and level of 1 hr accumulation in forecast files
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

# input and output data directories for each application in PROCESS_LIST
OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_WeatherRegime/ERA/
↪Regrid
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_WeatherRegime/ERA/
↪Daily

# 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 = Z500_daily_{valid?fmt=%Y%m%d?shift=-64800}_NH.nc
```

(continues on next page)

(continued from previous page)

```
# 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
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
```

(continues on next page)

(continued from previous page)

```
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/UserScript_fcstGFS_obsERA_WeatherRegime

# Directory to send output plots
WR_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s/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
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/UserScript_fcstGFS_obsERA_WeatherRegime/mpr
```

(continues on next page)

(continued from previous page)

```
# Run the Weather Regime Script
[script_wr]
# Timing Information
LEAD_SEQ = 24

# Run the user script once
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/UserScript_fcstGFS_obsERA_
↪WeatherRegime/ERA/Daily/Z500_daily_{valid?fmt=%Y%m%d}_NH.nc,{INPUT_BASE}/model_
↪applications/s2s/UserScript_fcstGFS_obsERA_WeatherRegime/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_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/UserScript_
↪fcstGFS_obsERA_WeatherRegime/WeatherRegime_driver.py


[sanal_wrclass]
# Format of VALID_BEG and VALID_END using % items
# Note, you cannot have hour, minutes, or seconds here
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d expands to YYYYMMDD
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 20001202

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 20170228

MODEL1 = GFS
MODEL1_OBTYPE = ADPUPA

# Location of MET config file to pass to StatAnalysis
```

(continues on next page)

(continued from previous page)

```
# References CONFIG_DIR from the [dir] section
STAT_ANALYSIS_CONFIG_FILE = {CONFIG_DIR}/STATAnalysisConfig_wrapped

# 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 MCTS -out_thresh >=1,>=2,>=3,>=4,>=5 -out_stat [out_
↪stat_file]

# Optional variables for further filtering
# can be blank, single, or multiple values
# if more than one use comma separated list
#
# (FCST)(OBS)_(VALID)(INIT)_HOUR_LIST: HH format (ex. 00, 06, 12)
# (FCST)(OBS)_LEAD_LIST: HH[H][MMSS] format (ex. 00, 06, 120)
MODEL_LIST = {MODEL1}
FCST_LEAD_LIST = 24
LINE_TYPE_LIST = MPR
# how to treat items listed in above _LIST variables
# GROUP_LIST_ITEMS: items listed in a given _LIST variable
#                   will be grouped together
# LOOP_LIST_ITEMS:  items listed in a give _LIST variable
#                   will be looped over
# if not listed METplus will treat the list as a group
GROUP_LIST_ITEMS = MODEL_LIST
LOOP_LIST_ITEMS = FCST_LEAD_LIST

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_WeatherRegime/
↪mpr/WeatherRegime

# Output data directory
STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_WeatherRegime

CONFIG_DIR = {PARM_BASE}/met_config

# Optional settings to create templated directory and file name information
# to save files as stat_analysis output as, this is appended to STAT_ANALYSIS_OUTPUT_DIR
# if no template is provided a default filename set in the code will be used
# Use:
# string templates can be set for all the lists being looped over, just
# use and a lower case version of the list, ex. {fcst_valid_hour?fmt=%H}
# or {fcst_var?fmt=%s}
# For looping over models:
```

(continues on next page)

(continued from previous page)

```
# can set MODELn_STAT_ANALYSIS_[DUMP_ROW/OUT_STAT]_TEMPLATE for individual models
# or STAT_ANALYSIS_[DUMP_ROW/OUT_STAT] with {model?fmt=%s}
MODEL1_STAT_ANALYSIS_OUT_STAT_TEMPLATE = {model?fmt=%s}_ERA_WRClass_{lead?fmt=%H%M%S}L_MCTS.
↪stat


[sanal_wrfreq]
# Format of VALID_BEG and VALID_END using % items
# Note, you cannot have hour, minutes, or seconds here
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d expands to YYYYMMDD
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 20001202

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 20170228

MODEL1 = GFS
MODEL1_OBTYPE = ADPUPA

# Location of MET config file to pass to StatAnalysis
# References CONFIG_DIR from the [dir] section
STAT_ANALYSIS_CONFIG_FILE = {CONFIG_DIR}/STATAnalysisConfig_wrapped

# 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 CNT -by DESC -out_stat [out_stat_file]

# Optional variables for further filtering
# can be blank, single, or multiple values
# if more than one use comma separated list
#
# (FCST)(OBS)_(VALID)(INIT)_HOUR_LIST: HH format (ex. 00, 06, 12)
# (FCST)(OBS)_LEAD_LIST: HH[H][MMSS] format (ex. 00, 06, 120)
MODEL_LIST = {MODEL1}
FCST_LEAD_LIST = 24
LINE_TYPE_LIST = MPR
# how to treat items listed in above _LIST variables
# GROUP_LIST_ITEMS: items listed in a given _LIST variable
```

(continues on next page)

(continued from previous page)

```
#                   will be grouped together
# LOOP_LIST_ITEMS:  items listed in a give _LIST variable
#                   will be looped over
# if not listed METplus will treat the list as a group
GROUP_LIST_ITEMS = MODEL_LIST
LOOP_LIST_ITEMS = FCST_LEAD_LIST

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_WeatherRegime/
↪mpr/freq

# Output data directory
STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_WeatherRegime

CONFIG_DIR = {PARM_BASE}/met_config

# Optional settings to create templated directory and file name information
# to save files as stat_analysis output as, this is appended to STAT_ANALYSIS_OUTPUT_DIR
# if no template is provided a default filename set in the code will be used
# Use:
# string templates can be set for all the lists being looped over, just
# use and a lower case version of the list, ex. {fcst_valid_hour?fmt=%H}
# or {fcst_var?fmt=%s}
# For looping over models:
# can set MODELn_STAT_ANALYSIS_[DUMP_ROW/OUT_STAT]_TEMPLATE for individual models
# or STAT_ANALYSIS_[DUMP_ROW/OUT_STAT] with {model?fmt=%s}
MODEL1_STAT_ANALYSIS_OUT_STAT_TEMPLATE = {model?fmt=%s}_ERA_WR_freq_{lead?fmt=%H%M%S}L_CNT.
↪stat
```

5.2.8.12.7 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

5.2.8.12.8 Python Scripts

This use case uses Python scripts to perform the blocking calculation

parm/use_cases/model_applications/s2s/UserScript_fcstGFS_obsERA_WeatherRegime/WeatherRegime_driver.py: 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')
```

(continues on next page)

(continued from previous page)

```
    # 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):
            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:]
```

(continues on next page)

(continued from previous page)

```
        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)

    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)
```

(continues on next page)

(continued from previous page)

```
        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)


    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()
```

(continues on next page)

(continued from previous page)

```
        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(',')
        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)
```

(continues on next page)

(continued from previous page)

```
    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).
↪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
```

(continues on next page)

(continued from previous page)

```
        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()
```

5.2.8.12.9 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/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/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
```

5.2.8.12.10 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/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.

5.2.8.12.11 Keywords

Note:

- RegridDataPlaneUseCase
- PCPCombineUseCase
- StatAnalysisUseCase
- S2SAppUseCase
- NetCDFFileUseCase
- GRIB2FileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

---

sphinx_gallery_thumbnail_path = '_static/s2s-OBS_ERA_weather_regime.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.8.13 UserScript: Make OMI plot from calculated MJO indices

model_applications/ s2s/ UserScript_fcstGFS_obsERA_OMI.py

5.2.8.13.1 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.

5.2.8.13.2 Datasets

- Forecast dataset: GFS Model Outgoing Longwave Radiation
- Observation dataset: ERA Reanlaysis Outgoing Longwave Radiation.

5.2.8.13.3 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

5.2.8.13.4 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.

5.2.8.13.5 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.

5.2.8.13.6 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/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.

```
# OMI UserScript wrapper
[config]
# 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 looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H
```

(continues on next page)

(continued from previous page)

```
# Start time for METplus run
VALID_BEG = 2017010100

# End time for METplus run
VALID_END = 2018123100

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 86400

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/use_cases/model_applications/s2s

# Run the obs for these cases
OBS_RUN = True
FCST_RUN = True

# Mask to use for regridding
REGRID_DATA_PLANE_VERIF_GRID = latlon 144 17 -20 0 2.5 2.5

# 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

# 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/UserScript_fcstGFS_obsERA_OMI/ERA/
↪Regrid
```

(continues on next page)

(continued from previous page)

```
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/UserScript_fcstGFS_obsERA_OMI/GFS/
↪Regrid
FCST_OLR_INPUT_TEMPLATE = OLR_{valid?fmt=%Y%m%d}.nc


# Configurations for pcp_combine:  Create daily means for the GFS
[daily_mean_fcst]
# run pcp_combine on obs data
FCST_PCP_COMBINE_RUN = {FCST_RUN}

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED
FCST_PCP_COMBINE_METHOD = USER_DEFINED

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/UserScript_fcstGFS_obsERA_OMI/GFS/daily_mean
FCST_PCP_COMBINE_OUTPUT_TEMPLATE = GFS_mean_{valid?fmt=%Y%m%d}.nc


# Configurations for regrid_data_plane:  Regrid ERA OLR to -20 to 20 latitude
[regrid_obs_olr]
LEAD_SEQ = 0
```

(continues on next page)

(continued from previous page)

```
# 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
OBS_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_NAME = olr

# Level of input field to process
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;

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = olr

# input and output data directories for each application in PROCESS_LIST
OBS_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/UserScript_fcstGFS_
↪obsERA_OMI/ERA/daily_mean
OBS_REGRID_DATA_PLANE_OUTPUT_DIR = {OBS_OLR_INPUT_DIR}

# format of filenames
# Input ERA Interim
OBS_REGRID_DATA_PLANE_INPUT_TEMPLATE = olr.1x.7920.nc
OBS_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {OBS_OLR_INPUT_TEMPLATE}


# Configurations for regrid_data_plane:  Regrid GFS OLR to -20 to 20 latitude
[regrid_fcst_olr]
# Run regrid_data_plane on forecast data
FCST_REGRID_DATA_PLANE_RUN = {FCST_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
FCST_REGRID_DATA_PLANE_VAR1_NAME = ULWRF_L0_mean

# Level of input field to process
FCST_REGRID_DATA_PLANE_VAR1_LEVELS = "(*,*)"
```

(continues on next page)

(continued from previous page)

```
# Name of output field to create
FCST_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = olr

# input and output data directories for each application in PROCESS_LIST
FCST_REGRID_DATA_PLANE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/UserScript_fcstGFS_
↪obsERA_OMI/GFS/daily_mean
FCST_REGRID_DATA_PLANE_OUTPUT_DIR = {FCST_OLR_INPUT_DIR}

# format of filenames
# Input ERA Interim
FCST_REGRID_DATA_PLANE_INPUT_TEMPLATE = GFS_mean_{valid?fmt=%Y%m%d}.nc
FCST_REGRID_DATA_PLANE_OUTPUT_TEMPLATE = {FCST_OLR_INPUT_TEMPLATE}


# 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/UserScript_fcstGFS_obsERA_
↪OMI/EOF/eof1/eof{valid?fmt=%j}.txt,{INPUT_BASE}/model_applications/s2s/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}
```

(continues on next page)

(continued from previous page)

```
# 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/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


# 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/UserScript_
↪fcstGFS_obsERA_OMI/OMI_driver.py
```

5.2.8.13.7 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.

5.2.8.13.8 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/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␣
↪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()
```

5.2.8.13.9 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/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/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
```

5.2.8.13.10 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/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/UserScript_fcstGFS_obsERA_OMI/plots (relative to **OUTPUT_BASE**).

5.2.8.13.11 Keywords

Note:

- S2SAppUseCase
- RegridDataPlaneUseCase
- PCPCombineUseCase

Navigate to *METplus Quick Search for Use Cases* (page 1595) to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/s2s-OMI_GFS_phase_diagram.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.8.14 Grid-Stat and Series-Analysis: BMKG APIK Seasonal Forecast

model_applications/s2s/GridStat_SeriesAnalysis _fcstNMME_obsCPC _seasonal_forecast.conf

5.2.8.14.1 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 ("validtime") 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.

5.2.8.14.2 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 Init-Time of hindcasts over past 29 years.

Hindcast Datasets:

Observational Dataset:

- CPC precipitation reference data (same format and grid)

5.2.8.14.3 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.

5.2.8.14.4 External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- netCDF4

5.2.8.14.5 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

5.2.8.14.6 METplus Configuration

```
# Grid to Grid APIK Verification - S2S Use Case 1: Comparison of NMME hindcasts to CPC␣
↪observations

[config]

# List of applications to run
PROCESS_LIST = GridStat, SeriesAnalysis(climo), SeriesAnalysis(full_stats)

# loop by INIT time (options are INIT, VALID, or ?)
LOOP_BY = INIT

# Format of INIT_BEG and INIT_END
INIT_TIME_FMT = %Y%m

# Start time for METplus run
INIT_BEG = 198207

# End time for METplus run
INIT_END = 201007
```

(continues on next page)

(continued from previous page)

```
INIT_INCREMENT = 1Y

# list of forecast leads to process  (JLV-NOTE: This only works for grid_stat and example␣
↪wrappers right now, using feature_281_py_embed)
LEAD_SEQ = 1m, 2m, 3m, 4m, 5m, 6m

#SERIES_ANALYSIS_CUSTOM_LOOP_LIST =

# Options are times, processes
# times = run all items in the PROCESS_LIST for a single initialization
# time, then repeat until all times have been evaluated.
# processes = run each item in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST.
LOOP_ORDER = processes

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

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 = "(*,*)"

# description of data to be processed
# used in output file path
MODEL = NMME
OBTYPE = CPC

# location of grid_stat MET config file
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

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
```

(continues on next page)

(continued from previous page)

```
# variables to describe format of forecast data
FCST_IS_PROB = false

# variables to describe format of observation data
#  none needed

# Increase verbosity of MET tools
#LOG_MET_VERBOSITY=4

GRID_STAT_OUTPUT_PREFIX = {MODEL}-hindcast_{CURRENT_OBS_NAME}_vs_{OBTYPE}_IC{init?fmt=%Y%b}_V
↪{valid?fmt=%Y%m%d}

# sets the desc variable in the SeriesAnalysis config file
SERIES_ANALYSIS_DESC = hindcast

# sets the cat_thresh variable in the SeriesAnalysis config file
SERIES_ANALYSIS_CAT_THRESH = >=50, >=100, >=150, >=200, >=250, >=300, >=400, >=500

# sets the vld_thresh variable in the SeriesAnalysis config file
SERIES_ANALYSIS_VLD_THRESH = 0.50

# sets the block_size variable in the SeriesAnalysis config file
SERIES_ANALYSIS_BLOCK_SIZE = 360*181

# set to True to add the -paired flag to the SeriesAnalysis command
SERIES_ANALYSIS_IS_PAIRED = False

# MET Configuration file passed to SeriesAnalysis
SERIES_ANALYSIS_CONFIG_FILE = {PARM_BASE}/met_config/SeriesAnalysisConfig_wrapped

# If True/yes, run plot_data_plane on output from Series-Analysis to generate
# images for each stat item listed in SERIES_ANALYSIS_STAT_LIST
SERIES_ANALYSIS_GENERATE_PLOTS = no

# If True/yes, run convert on output from Series-Analysis to generate
# a gif using images in groups of name/level/stat
SERIES_ANALYSIS_GENERATE_ANIMATIONS = no

# grid to remap data. Value is set as the 'to_grid' variable in the 'regrid' dictionary
# See MET User's Guide for more information
#SERIES_ANALYSIS_REGRID_TO_GRID = NONE

SERIES_ANALYSIS_RUNTIME_FREQ = RUN_ONCE_PER_LEAD

SERIES_ANALYSIS_RUN_ONCE_PER_STORM_ID = False
```

(continues on next page)

(continued from previous page)

```
# used for SeriesAnalysis(climo) instance
SERIES_ANALYSIS_STAT_LIST = OBAR

[full_stats]

SERIES_ANALYSIS_STAT_LIST =TOTAL, FBAR, OBAR, ME, MAE, RMSE, ANOM_CORR, PR_CORR
SERIES_ANALYSIS_CTS_LIST = BASER, CSI, GSS

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


[dir]

# input and output data directories
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/NMME/hindcast/monthly
OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/NMME/obs
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}

SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/s2s/GridStat_SeriesAnalysis_
↪fcstNMME_obsCPC_seasonal_forecast/SeriesAnalysis

# used in full_stats instance file only
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_DIR =


[filename_templates]

# format of filenames
# FCST
FCST_GRID_STAT_INPUT_TEMPLATE = nmme_pr_hcst_{init?fmt=%b}IC_{valid?fmt=%m}_*.nc

# ANLYS
OBS_GRID_STAT_INPUT_TEMPLATE = obs_cpc_pp.1x1.nc

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_TEMPLATE = series_analysis_{MODEL}_{OBTYPE}_stats_F{lead?fmt=%2m}_
↪{instance?fmt=%s}.nc
```

(continues on next page)

(continued from previous page)

```
# used in full_stat instance only
SERIES_ANALYSIS_CLIMO_MEAN_INPUT_TEMPLATE =
```

5.2.8.14.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

GridStatConfig_wrapped

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//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 207) 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 = {
   grid = "";
   poly = "";
}
```

(continues on next page)

(continued from previous page)

```
//
// 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}
```

5.2.8.14.8 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 Grid-

Stat_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.

5.2.8.14.9 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_seaso
(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
```

5.2.8.14.10 Keywords

Note:

- GridStatToolUseCase
- SeriesAnalysisUseCase
- NetCDFFileUseCase
- LoopByMonthFeatureUseCase
- NCAROrgUseCase
- RuntimeFreqUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.8.15 Blocking Calculation: GFS and ERA RegridDataPlane, PcpCombine, and Blocking python code

model_applications/ s2s/ UserScript_fcstGFS_obsERA_Blocking.py

5.2.8.15.1 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.

5.2.8.15.2 Datasets

- Forecast dataset: GFS Forecast 500 mb height.
- Observation dataset: ERA Reanlaysis 500 mb height.

5.2.8.15.3 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

5.2.8.15.4 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

5.2.8.15.5 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.

5.2.8.15.6 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/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.

```
# UserScript wrapper example

[config]
# List of applications to run - Pre-Processing and Blocking Script
# 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)
# List of applications to run - Omit Pre-Processing Steps
PROCESS_LIST = UserScript(create_cbl_filelist), UserScript(script_blocking),␣
↪StatAnalysis(sanal_ibls), StatAnalysis(sanal_blocks)

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 2000120100

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 2017022800

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 86400

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0
```

(continues on next page)

(continued from previous page)

```
# Only Process DJF
SKIP_TIMES = "%m:begin_end_incr(3,11,1)", "%m%d:0229"

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = processes

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/use_cases/model_applications/s2s

OBS_ANOM_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/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/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

FCST_AVE_INPUT_DIR = {INPUT_BASE}/model_applications/s2s/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


# Forecast Regridding to 1 degree using regrid_data_plane
[regrid_fcst]
# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = INIT

# Format of INIT_BEG and INIT_END
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run
```

(continues on next page)

(continued from previous page)

```
INIT_BEG = 2000120100

# End time for METplus run
INIT_END = 2017022800

# Increment between METplus runs in seconds. Must be >= 60
INIT_INCREMENT = 86400

# list of forecast leads to process
LEAD_SEQ = 24

# REGRID_DATA_PLANE (Step 1)
# Run regrid_data_plane on forecast data
FCST_REGRID_DATA_PLANE_RUN = True

# If true, process each field individually and write a file for each
# If false, run once per run time passing in all fields specified
FCST_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
FCST_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Z500

# Level of input field to process
FCST_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = P500

# Name of output field to create
FCST_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = Z500

# Mask to use for regridding
REGRID_DATA_PLANE_VERIF_GRID = latlon 360 90 89 0 -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

# input and output data directories for each application in PROCESS_LIST
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/UserScript_fcstGFS_obsERA_Blocking/
↪FV3GFS/Regrid

# format of filenames
# Input ERA Interim
FCST_REGRID_DATA_PLANE_INPUT_TEMPLATE = {init?fmt=%Y%m%d%H}/gfs.t00z.pgrb2.0p25.f{lead?fmt=
↪%HHH}
```

(continues on next page)

(continued from previous page)

```
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


# Observation Regridding to 1 degree using regrid_data_plane
[regrid_obs]
# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = VALID

# Format of INIT_BEG and INIT_END
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 1979120100

# End time for METplus run
VALID_END = 2017022818

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 21600

# list of forecast leads to process
LEAD_SEQ = 0

# Run regrid_data_plane on forecast data
OBS_REGRID_DATA_PLANE_RUN = True

# If true, process each field individually and write a file for each
# If false, run once per run time passing in all fields specified
OBS_DATA_PLANE_ONCE_PER_FIELD = False

# Name of input field to process
OBS_REGRID_DATA_PLANE_VAR1_INPUT_FIELD_NAME = Z

# Level of input field to process
OBS_REGRID_DATA_PLANE_VAR1_INPUT_LEVEL = P500

# Name of output field to create
OBS_REGRID_DATA_PLANE_VAR1_OUTPUT_FIELD_NAME = Z500

# Mask to use for regridding
REGRID_DATA_PLANE_VERIF_GRID = latlon 360 90 89 0 -1.0 1.0

# Method to run regrid_data_plane, not setting this will default to NEAREST
REGRID_DATA_PLANE_METHOD = BILIN
```

(continues on next page)

(continued from previous page)

```
# Regridding width used in regrid_data_plane, not setting this will default to 1
REGRID_DATA_PLANE_WIDTH = 2

# input and output data directories for each application in PROCESS_LIST
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/UserScript_fcstGFS_obsERA_Blocking/ERA/
↪Regrid

# format of filenames
# Input ERA Interim
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


# Perform a sum over the 4 daily times that have been regridded using pcp_combine
# 00, 06, 12, 18 UTC
[daily_mean_obs]
# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = VALID

# Format of INIT_BEG and INIT_END
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 1979120118

# End time for METplus run
VALID_END = 2017022818

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 86400

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = True

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, and DERIVE
OBS_PCP_COMBINE_METHOD = DERIVE
OBS_PCP_COMBINE_STAT_LIST = MEAN

# field name and level of 1 hr accumulation in forecast files
OBS_PCP_COMBINE_INPUT_ACCUMS = 6
```

(continues on next page)

(continued from previous page)

```
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 output and set 24 hours as the accumulation
OBS_PCP_COMBINE_OUTPUT_NAME = Z500
OBS_PCP_COMBINE_OUTPUT_ACCUM = 24
OBS_PCP_COMBINE_DERIVE_LOOKBACK = 24

# input and output data directories for each application in PROCESS_LIST
OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s/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}


# Perform a 5 day running mean on the data using pcp_combine
[running_mean_obs]
# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = VALID

# Format of INIT_BEG and INIT_END
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 1979120100

# End time for METplus run
VALID_END = 2017022800

# Increment between METplus runs in seconds. Must be >= 60
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"

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = TRUE

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, and DERIVE
OBS_PCP_COMBINE_METHOD = DERIVE
```

(continues on next page)

(continued from previous page)

```
OBS_PCP_COMBINE_STAT_LIST = MEAN

# field name, level and setting time attribute of 1 hr accumulation in forecast files
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}";

# Set output variable name
OBS_PCP_COMBINE_OUTPUT_NAME = Z500

#  Running mean is 5 days
OBS_PCP_COMBINE_OUTPUT_ACCUM = 120
OBS_PCP_COMBINE_DERIVE_LOOKBACK = 120

# input and output data directories for each application in PROCESS_LIST
OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking/ERA/Daily
OBS_PCP_COMBINE_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking/ERA/Rmean5d

# format of filenames
# Input ERA Interim
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


# Compute anomalies using the daily means and 5 day running mean using pcp_combine
[anomaly_obs]
# time looping - options are INIT, VALID, RETRO, and REALTIME
LOOP_BY = VALID

# Format of INIT_BEG and INIT_END
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run
VALID_BEG = 1979120100

# End time for METplus run
VALID_END = 2017022800

# Increment between METplus runs in seconds. Must be >= 60
VALID_INCREMENT = 86400

# list of forecast leads to process
LEAD_SEQ = 0
```

(continues on next page)

(continued from previous page)

```
# 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"

# run pcp_combine on obs data
OBS_PCP_COMBINE_RUN = True

# method to run pcp_combine on forecast data
# Options are ADD, SUM, SUBTRACT, DERIVE, and USER_DEFINED
OBS_PCP_COMBINE_METHOD = USER_DEFINED

# User defined pcp_combine command
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="(*,*)";'

# input and output data directories for each application in PROCESS_LIST
OBS_PCP_COMBINE_INPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking/ERA
OBS_PCP_COMBINE_OUTPUT_DIR = {OBS_ANOM_INPUT_DIR}

# format of filenames
# Input ERA Interim
OBS_PCP_COMBINE_INPUT_TEMPLATE = Z500_daily_{valid?fmt=%Y%m%d}_NH.nc
OBS_PCP_COMBINE_OUTPUT_TEMPLATE = {OBS_ANOM_INPUT_TEMPLATE}


# 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
LEAD_SEQ = 0

# Template of filenames to input to the user-script
USER_SCRIPT_INPUT_TEMPLATE = {INPUT_BASE}/model_applications/s2s/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
```

(continues on next page)

(continued from previous page)

```
# 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
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
```

(continues on next page)

(continued from previous page)

```
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/UserScript_fcstGFS_obsERA_Blocking/mpr

# Plots Output Dir
BLOCKING_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/s2s/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

# IBL plot legend for forecast and obs
IBL_PLOT_OBS_LABEL = ERA Reanalysis
IBL_PLOT_FCST_LABEL = GEFS


# Run the Blocking Analysis Script
[script_blocking]
```

(continues on next page)

(continued from previous page)

```
# 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/UserScript_fcstGFS_obsERA_
↪Blocking/ERA/Daily/Z500_daily_{valid?fmt=%Y%m%d}_NH.nc,{INPUT_BASE}/model_applications/s2s/
↪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/UserScript_
↪fcstGFS_obsERA_Blocking/Blocking_driver.py


# Stat Analysis for the IBLs
[sanal_ibls]
# Format of VALID_BEG and VALID_END using % items
# Note, you cannot have hour, minutes, or seconds here
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d expands to YYYYMMDD
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 20001201

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 20170228

MODEL1 = GFS
MODEL1_OBTYPE = ADPUPA

# Location of MET config file to pass to StatAnalysis
# References CONFIG_DIR from the [dir] section
```

(continues on next page)

(continued from previous page)

```
STAT_ANALYSIS_CONFIG_FILE = {CONFIG_DIR}/STATAnalysisConfig_wrapped

# 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]

# Optional variables for further filtering
# can be blank, single, or multiple values
# if more than one use comma separated list
#
# (FCST)(OBS)_(VALID)(INIT)_HOUR_LIST: HH format (ex. 00, 06, 12)
# (FCST)(OBS)_LEAD_LIST: HH[H][MMSS] format (ex. 00, 06, 120)
MODEL_LIST = {MODEL1}
FCST_LEAD_LIST = 24
LINE_TYPE_LIST = MPR
# how to treat items listed in above _LIST variables
# GROUP_LIST_ITEMS: items listed in a given _LIST variable
#                   will be grouped together
# LOOP_LIST_ITEMS:  items listed in a give _LIST variable
#                   will be looped over
# if not listed METplus will treat the list as a group
GROUP_LIST_ITEMS = MODEL_LIST
LOOP_LIST_ITEMS = FCST_LEAD_LIST

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking/mpr/
↪IBL

# Output data directory
STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking

CONFIG_DIR = {PARM_BASE}/met_config

# Optional settings to create templated directory and file name information
# to save files as stat_analysis output as, this is appended to STAT_ANALYSIS_OUTPUT_DIR
# if no template is provided a default filename set in the code will be used
# Use:
# string templates can be set for all the lists being looped over, just
# use and a lower case version of the list, ex. {fcst_valid_hour?fmt=%H}
# or {fcst_var?fmt=%s}
# For looping over models:
# can set MODELn_STAT_ANALYSIS_[DUMP_ROW/OUT_STAT]_TEMPLATE for individual models
# or STAT_ANALYSIS_[DUMP_ROW/OUT_STAT] with {model?fmt=%s}
```

(continues on next page)

(continued from previous page)

```
MODEL1_STAT_ANALYSIS_OUT_STAT_TEMPLATE = {model?fmt=%s}_ERA_IBLS_{lead?fmt=%H%M%S}L_CTS_CNT.
↪stat


# Stat Analysis for the Blocks
[sanal_blocks]
# Format of VALID_BEG and VALID_END using % items
# Note, you cannot have hour, minutes, or seconds here
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d expands to YYYYMMDD
VALID_TIME_FMT = %Y%m%d

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 20001201

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 20170228

MODEL1 = GFS
MODEL1_OBTYPE = ADPUPA

# Location of MET config file to pass to StatAnalysis
# References CONFIG_DIR from the [dir] section
STAT_ANALYSIS_CONFIG_FILE = {CONFIG_DIR}/STATAnalysisConfig_wrapped

# 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]

# Optional variables for further filtering
# can be blank, single, or multiple values
# if more than one use comma separated list
#
# (FCST)(OBS)_(VALID)(INIT)_HOUR_LIST: HH format (ex. 00, 06, 12)
# (FCST)(OBS)_LEAD_LIST: HH[H][MMSS] format (ex. 00, 06, 120)
MODEL_LIST = {MODEL1}
FCST_LEAD_LIST = 24
LINE_TYPE_LIST = MPR
# how to treat items listed in above _LIST variables
# GROUP_LIST_ITEMS: items listed in a given _LIST variable
#                   will be grouped together
```

(continues on next page)

(continued from previous page)

```
# LOOP_LIST_ITEMS:  items listed in a give _LIST variable
#                   will be looped over
# if not listed METplus will treat the list as a group
GROUP_LIST_ITEMS = MODEL_LIST
LOOP_LIST_ITEMS = FCST_LEAD_LIST

MODEL1_STAT_ANALYSIS_LOOKIN_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking/mpr/
↪Blocks

# Output data directory
STAT_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/s2s/UserScript_fcstGFS_obsERA_Blocking

CONFIG_DIR = {PARM_BASE}/met_config

# Optional settings to create templated directory and file name information
# to save files as stat_analysis output as, this is appended to STAT_ANALYSIS_OUTPUT_DIR
# if no template is provided a default filename set in the code will be used
# Use:
# string templates can be set for all the lists being looped over, just
# use and a lower case version of the list, ex. {fcst_valid_hour?fmt=%H}
# or {fcst_var?fmt=%s}
# For looping over models:
# can set MODELn_STAT_ANALYSIS_[DUMP_ROW/OUT_STAT]_TEMPLATE for individual models
# or STAT_ANALYSIS_[DUMP_ROW/OUT_STAT] with {model?fmt=%s}
MODEL1_STAT_ANALYSIS_OUT_STAT_TEMPLATE = {model?fmt=%s}_ERA_Blocks_{lead?fmt=%H%M%S}L_CTS.
↪stat
```

5.2.8.15.7 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

5.2.8.15.8 Python Scripts

This use case uses Python scripts to perform the blocking calculation

parm/use_cases/model_applications/s2s/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,
↪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,
```

(continues on next page)

(continued from previous page)

```
            '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):
            raise Exception('Must run observed IBLs before running GIBLs.')
```

(continues on next page)

(continued from previous page)

```
        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.')
        print('Plotting Obs Blocks')
```

(continues on next page)

(continued from previous page)

```
        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()
```

5.2.8.15.9 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/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/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
```

5.2.8.15.10 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/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.

5.2.8.15.11 Keywords

Note:

- RegridDataPlaneUseCase
- PCPCombineUseCase
- StatAnalysisUseCase
- S2SAppUseCase
- NetCDFFileUseCase
- GRIB2FileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

sphinx_gallery_thumbnail_path = '_static/s2s-IBL_frequency.png'

Total running time of the script: (0 minutes 0.000 seconds)

5.2.9 Space Weather

Upper atmosphere and geospace model configurations

5.2.9.1 GenVxMask: Solar Altitude

model_applications/space_weather/GenVxMask_fcstGloTEC_solar _altitude.conf

5.2.9.1.1 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]

5.2.9.1.2 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.

5.2.9.1.3 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 1479) section for more information.

Data source: NOAA Space Weather Prediction Center (SWPC)
Data contact: Dominic Fuller-Rowell (dominic.fuller-rowell@noaa.gov)

5.2.9.1.4 METplus Use Case Contact

Author: Jonathan L. Vigh (National Center for Atmospheric Research / Research Applications Laboratory / Joint Numerical Testbed)
Last modified: 26 May 2020

5.2.9.1.5 METplus Components

This use case utilizes the METplus GenVxMask wrapper to generate a command to run the MET tool GenVx-Mask if all required files are found.

5.2.9.1.6 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.

5.2.9.1.7 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/space_weather/GenVxMask_fcstGloTEC_FcstOnly_solar_altitude.conf

```
# GenVxMask METplus Configuration for the GenVxMask_glotec_solar_altitude space weather use␣
↪case:
#    Creating masks based on the solar altitude angle for GloTEC model data (id: vx12)
#
# Author: Jonathan Vigh (NCAR/RAL/JNTP)
#
# Description: 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. Patricks 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.
```

(continues on next page)

(continued from previous page)

```
#               [Source: https://sciencing.com/solar-altitude-23364.html]
#

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]
## Configuration-related settings such as the process list, begin and end times, etc.

# List of applications to run - only GenVxMask for this case
PROCESS_LIST = GenVxMask

# time looping - 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
LOOP_BY = VALID

# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H%M

# Start time for METplus run - must match INIT_TIME_FMT
VALID_BEG = 201503170005

# End time for METplus run - must match INIT_TIME_FMT
VALID_END = 201503170015

# Just run the first two time points for this use case example
# replace with 201503172355 process the entire day

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 600

# List of forecast leads to process for each run time (init or valid)
# If unset, defaults to 0 (don't loop through forecast leads
LEAD_SEQ = 0

# The above configuration will loop by valid time in increments of
#  VALID_INCREMENT from VALID_BEG to VALID_END. Since LEAD_SEQ is set to 0,
#  it will not loop over any forecast lead times.
#  This will run:
```

(continues on next page)

(continued from previous page)

```
#  Valid: 2015-03-17_0005Z Forecast lead: 0
#      to 2015-03-17_0055Z Forecast lead: 0


# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for GenVxMask only
LOG_GEN_VX_MASK_VERBOSITY = 2

GEN_VX_MASK_SKIP_IF_OUTPUT_EXISTS = False

# Time relative to valid time (in seconds if no units are specified) to allow files to be␣
↪considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
GEN_VX_MASK_FILE_WINDOW_BEGIN = 0
GEN_VX_MASK_FILE_WINDOW_END = 0

# Command line arguments to add to the call to gen_vx_mask
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;'

[filename_templates]

# Template to look for input to GenVxMask relative to GEN_VX_MASK_INPUT_DIR
GEN_VX_MASK_INPUT_TEMPLATE = GloTEC_TEC_{valid?fmt=%Y_%m_%d}_cosmic.nc

GEN_VX_MASK_INPUT_MASK_TEMPLATE = GloTEC_TEC_{valid?fmt=%Y_%m_%d}_cosmic.nc

# Template to use to write output from GenVxMask
GEN_VX_MASK_OUTPUT_TEMPLATE = GloTEC_TEC_solar_altitude_le_45_masked_{valid?fmt=%Y_%m_%d_%H
↪%M}.nc


[dir]
```

(continues on next page)

(continued from previous page)

```
# Input/Output directories can be left empty if the corresponding template contains the full␣
↪path to the files
GEN_VX_MASK_INPUT_DIR = {INPUT_BASE}/model_applications/space_weather/glotec_vs_glotec/GLO_
↪20190422_with_cosmic

GEN_VX_MASK_INPUT_MASK_DIR = {INPUT_BASE}/model_applications/space_weather/glotec_vs_glotec/
↪GLO_20190422_with_cosmic

GEN_VX_MASK_OUTPUT_DIR={OUTPUT_BASE}/model_applications/space_weather/GenVxMask_glotec_solar_
↪altitude
```

5.2.9.1.8 MET Configuration

None. GenVxMask does not use configuration files.

5.2.9.1.9 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
```

(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.

5.2.9.1.10 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

5.2.9.1.11 Keywords

Note:

- GenVxMaskToolUseCase
- SpaceWeatherAppUseCase
- NOAASWPCOrgUseCase
- MaskingFeatureUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.9.2 Grid-Stat: Analysis validation

GridStat_fcstGloTEC _obsGloTEC_vx7.conf

5.2.9.2.1 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

5.2.9.2.2 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.

5.2.9.2.3 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 1492) section for more information.
Data source: NOAA Space Weather Prediction Center (SWPC)
Data contact: Dominic Fuller-Rowell (dominic.fuller-rowell@noaa.gov)

5.2.9.2.4 METplus Use Case Contact

Author: Jonathan L. Vigh (National Center for Atmospheric Research / Research Applications Laboratory / Joint Numerical Testbed)
Last modified: 06 February 2020

5.2.9.2.5 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.

5.2.9.2.6 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

5.2.9.2.7 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/space_weather/GridStat_fcstGloTEC_obsGloTEC_vx7.conf

```
# GridStat METplus Configuration for the glotec_vs_glotec space weather use case:
#    GloTEC initialized with and without COSMIC-1 RO data (id: vx7)
#
# Author: Jonathan Vigh (NCAR/RAL/JNTP)
#
# Description: 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.
```

(continues on next page)

(continued from previous page)

```
#              The observations are considered to be the run with COSMIC-1 RO data.
#
#              This use case runs grid_stat for all of the forecast times for one day for a
#              space weather event known as the St. Patricks Day Storm (Mar 17, 2015).
#
#              Novel aspects of this use case:
#                - 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 value of a quality flag at each time step␣
↪and grid point
#
#

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# 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

# List of applications to run - only GridStat for this case
PROCESS_LIST = GridStat

# time looping - 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
LOOP_BY = VALID
```

(continues on next page)

(continued from previous page)

```
# Format of VALID_BEG and VALID_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H%M

# Start time for METplus run - must match VALID_TIME_FMT
VALID_BEG = 201503170005

# End time for METplus run - must match VALID_TIME_FMT
VALID_END = 201503170015
# Just run the first two time points for this use case example
# replace with 201503172355 process the entire day

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 600

# List of forecast leads to process for each run time (init or valid)
LEAD_SEQ = 0

# The above configuration will loop by valid time in increments of
#  VALID_INCREMENT from VALID_BEG to VALID_END. Since LEAD_SEQ is set to 0,
#  it will not loop over any forecast lead times.
#  This will run:
#  Valid: 2015-03-17_0005Z Forecast lead: 0
#      to 2015-03-17_0055Z Forecast lead: 0

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for GridStat only
#LOG_GRID_STAT_VERBOSITY = 2

# Location of MET config file to pass to the GridStat
GRID_STAT_CONFIG_FILE = {PARM_BASE}/met_config/GridStatConfig_wrapped

# Override MET config file settings for this use case
```

(continues on next page)

(continued from previous page)

```
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

# Name to identify model (forecast) data in output
MODEL = GloTEC_without_cosmic

# Name to identify observation data in output (used in output file path)
OBTYPE = GloTEC_with_cosmic

# List of variables to compare in GridStat - FCST_VAR1 variables correspond
#  to OBS_VAR1 variables

# Name of forecast variable 1
BOTH_VAR1_NAME = TEC

# List of levels to evaluate for forecast variable 1
# NOTE: this uses the new capability in METplus v3.0 to specify levels with valid time
# Previously, a user would have had to provide a list, such as:
# FCST_VAR1_LEVELS = "(20150317_000500,*,*)", "(20150317_001500,*,*)", "( 20150317_002500,*,
↪*)", "( 20150317_003500,*,*)", "( 20150317_004500,*,*)"
BOTH_VAR1_LEVELS = "({valid?fmt=%Y%m%d_%H%M%S},*,*)"

# NOTE that if the values do not match exactly, one can specify a time offset, as follows:
#FCST_VAR1_LEVELS = "({valid?fmt=%Y%m%d_%H%M%S?shift=5M},*,*)"

# List of thresholds to evaluate for each name/level combination for
#  forecast variable 1
# Not used for this example
#FCST_VAR1_THRESH = gt10.0, gt20.0, gt30.0, gt40.0

# Name of observation variable 1 (this is specified in the GridStat.conf file)
# Not used for this example
#OBS_VAR1_NAME = APCP_03

# List of levels to evaluate for observation variable 1
# (*,*) is NetCDF notation - must include quotes around these values!
```

(continues on next page)

(continued from previous page)

```
# must be the same lenght as FCST_VAR1_LEVELS
# Not used for this example
#OBS_VAR1_LEVELS = "(*,*)"

# List of thresholds to evaluate for each name/level combination for
#  forecast variable 1 - must be the same length as FCST_VAR1_THRESH
# Not used for this example
#OBS_VAR1_THRESH = gt10.0, gt20.0, gt30.0, gt40.0

# Time relative to valid time (in seconds) to allow files to be considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
#  Not used in this example.
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

# MET GridStat neighborhood values
# See the MET User's Guide GridStat section for more information

# width value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_WIDTH = 1

# shape value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

# Set to true to run GridStat separately for each field specified
# Set to false to create one run of GridStat per run time that
#   includes all fields specified.
# Not used for this example
GRID_STAT_ONCE_PER_FIELD = False

# Set to true if forecast data is probabilistic
FCST_IS_PROB = false

# Only used if FCST_IS_PROB is true - sets probabilistic threshold
# Not used for this example
FCST_GRID_STAT_PROB_THRESH = ==0.1

# Set to true if observation data is probabilistic
#  Only used if configuring forecast data as the 'OBS' input
OBS_IS_PROB = false

# Only used if OBS_IS_PROB is true - sets probabilistic threshold
# Not used for this example
```

(continues on next page)

(continued from previous page)

```
OBS_GRID_STAT_PROB_THRESH = ==0.1

# Output prefix set in grid_stat config file
GRID_STAT_OUTPUT_PREFIX={MODEL}-vx7_{CURRENT_OBS_NAME}_vs_{OBTYPE}

GRID_STAT_DESC = vx7

# End of [config] section and start of [dir] section
[dir]

# directory containing forecast input to GridStat
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/space_weather/glotec_vs_glotec/
↪GLO_20190422_without_cosmic

# directory containing observation input to GridStat
OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/space_weather/glotec_vs_glotec/GLO_
↪20190422_with_cosmic

# directory containing climatology input to GridStat
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_DIR =

# directory to write output from GridStat
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/space_weather/glotec_vs_glotec

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to GridStat relative to FCST_GRID_STAT_INPUT_DIR
FCST_GRID_STAT_INPUT_TEMPLATE = GloTEC_TEC_{valid?fmt=%Y_%m_%d}.nc

# Template to look for observation input to GridStat relative to OBS_GRID_STAT_INPUT_DIR
OBS_GRID_STAT_INPUT_TEMPLATE = GloTEC_TEC_{valid?fmt=%Y_%m_%d}_cosmic.nc

# Optional subdirectories relative to GRID_STAT_OUTPUT_DIR to write output from GridStat
GRID_STAT_OUTPUT_TEMPLATE = {valid?fmt=%Y_%m_%d}

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_MEAN_INPUT_
↪DIR
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

# Used to specify one or more verification mask files for GridStat
# Not used for this example
GRID_STAT_VERIFICATION_MASK_TEMPLATE =
```

5.2.9.2.8 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *GridStat MET Configuration* (page 135) 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}

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

//grid_weight_flag =
${METPLUS_GRID_WEIGHT_FLAG}

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.9.2.9 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.

5.2.9.2.10 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

5.2.9.2.11 Keywords

Note:

- GridStatToolUseCase
- SpaceWeatherAppUseCase
- NOAASWPCOrgUseCase
- CustomStringLoopingUseCase
- MaskingFeatureUseCase
- ValidationUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.10 Tropical Cyclone and Extra Tropical Cyclone

Any field that is associated with Tropical Cyclone and Extra-tropical Cyclones

5.2.10.1 TCRMW: Hurricane Gonzalo

model_applications/tc_and_extra_tc/TCRMW_fcstGFS_fcstOnly _gonzolo.conf

5.2.10.1.1 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.

5.2.10.1.2 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 1500) section for more information.

5.2.10.1.3 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.

5.2.10.1.4 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

5.2.10.1.5 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

```
# TCRMW Gonzalo - METplus Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# List of applications to run - only TCRMW for this case
PROCESS_LIST = TCRMW

# time looping - 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
LOOP_BY = INIT

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
INIT_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
INIT_BEG = 2014101312

# End time for METplus run - must match INIT_TIME_FMT
INIT_END = 2014101312

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
INIT_INCREMENT = 6H

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
# begin_end_incr(x,y,z) expands to a list where
# x = begin, y = end (inclusive), and z = increment between each value
#LEAD_SEQ = begin_end_incr(0, 126, 6)
LEAD_SEQ = begin_end_incr(0, 24, 6)

# Order of loops to process data - Options are times, processes
```
(continues on next page)

(continued from previous page)

```
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times

# Verbosity of MET output - overrides LOG_VERBOSITY for TCRMW only
LOG_TC_RMW_VERBOSITY = 3

# Location of MET config file to pass to TCRMW
# References CONFIG_DIR from the [dir] section
TC_RMW_CONFIG_FILE = {CONFIG_DIR}/TCRMWConfig_wrapped

# type of data used for input to TCRMW data dictionary
# The value set here will be used to add 'data_type = <value>;'
# If this option is removed/commented out/empty, nothing will be set
TC_RMW_INPUT_DATATYPE = GRIB2

MODEL = HCLT

# list of pressure values to be referenced by other config variables
# this is not a variable name known to METplus, but added to avoid repeating values
PRESSURE_LEVELS = "P1000","P850","P700","P500","P300","P200","P150","P100"

# List of variables to process in TCRMW
# must use BOTH_ config variables regardless if input is forecast or observation
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
BOTH_VAR4_LEVELS = {PRESSURE_LEVELS}

BOTH_VAR5_NAME = UGRD
BOTH_VAR5_LEVELS = {PRESSURE_LEVELS}

BOTH_VAR5_NAME = VGRD
```

(continues on next page)

(continued from previous page)

```
BOTH_VAR5_LEVELS = {PRESSURE_LEVELS}

# The following variables set MET configuration variables of the same name,
# i.e. TC_RMW_BASIN sets basin, TC_RMW_STORM_NAME sets storm_name, etc.
TC_RMW_BASIN =

TC_RMW_STORM_ID =

TC_RMW_CYCLONE =

# Regrid options in TCRMW config file
# If these options are removed/commented out/empty, they will use
# the values from default MET config file
TC_RMW_REGRID_METHOD = BILIN

TC_RMW_REGRID_WIDTH = 2

TC_RMW_REGRID_VLD_THRESH = 0.5

TC_RMW_REGRID_SHAPE = SQUARE

# The following variables set values in the MET
# configuration file used by this example
# Leaving these values commented will use the value
# found in the default MET configuration file
#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 =


# Other TCRMW config file options
# If these options are removed/commented out/empty, they will use
# the values from default MET config file

#TC_RMW_N_RANGE = 100

#TC_RMW_N_AZIMUTH = 180

#TC_RMW_MAX_RANGE_KM = 1000.0

#TC_RMW_DELTA_RANGE_KM = 10.0
```

(continues on next page)

(continued from previous page)

```
#TC_RMW_SCALE = 0.2


#
#  DIRECTORIES
#
[dir]

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/met_config

# Location of input track data directory
# for DECK data
TC_RMW_DECK_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/rmw/adeck

# directory containing input data files
TC_RMW_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/rmw/fcst

# directory to write output files
TC_RMW_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/tc_and_extra_tc/TCRMW_gonzalo

[filename_templates]

# template to use to find DECK files relative to TC_RMW_DECK_INPUT_DIR
TC_RMW_DECK_TEMPLATE = gonzalo08l.{init?fmt=%Y%m%d%H}.f00-24.trak.hwrf.atcfunix.06hr

# template to use to find input files relative to TC_RMW_INPUT_DIR
TC_RMW_INPUT_TEMPLATE = gonzalo08l.subset.{init?fmt=%Y%m%d%H}.hwrfprs.core.0p02.f{lead?fmt=
↪%3H}.grb2

# template to use write output files relative to TC_RMW_OUTPUT_DIR
TC_RMW_OUTPUT_TEMPLATE = tc_rmw_gonzal09l.{init?fmt=%Y%m%d%H}.nc
```

5.2.10.1.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *TCRMW MET Configuration* (page 250) 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}
```

5.2.10.1.7 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.

5.2.10.1.8 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

5.2.10.1.9 Keywords

Note:

- TCRMWToolUseCase
- GRIB2FileUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.10.2 Track and Intensity Plotter: Generate mean, median and box plots

model_applications/tc_and_extra_tc/Plotter_fcstGFS _obsGFS_RPlotting.conf

5.2.10.2.1 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.

5.2.10.2.2 Datasets

- Forecast dataset: ADeck ATCF tropical cyclone data
- Observation dataset: BDeck ATCF tropical cyclone "best track" cyclone data

5.2.10.2.3 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.

5.2.10.2.4 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).

5.2.10.2.5 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

```
#
#   PRECONDITION: REQUIRES INSTALLATION OF R on user system
#

#
#  CONFIGURATION
#
[config]

# Loop over each process in the process list (set in PROCESS_LIST) for all times in the time␣
↪window of
# interest.
LOOP_ORDER = processes
# Configuration files
TC_PAIRS_CONFIG_FILE = {PARM_BASE}/met_config/TCPairsConfig_wrapped

PROCESS_LIST = TCPairs, TCMPRPlotter


# The init time begin and end times, increment
LOOP_BY = INIT
INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20141214
INIT_END = 20141214

# This is the step-size. Increment in seconds from the begin time to the end
# time
INIT_INCREMENT = 21600 ;; set to every 6 hours=21600 seconds

# A list of times to include, in format YYYYMMDD_hh
INIT_INCLUDE =

# A list of times to exclude, in format YYYYMMDD_hh
INIT_EXCLUDE =

#
# Specify model valid time window in format YYYYMM[DD[_hh]].  Only tracks
# that fall within the valid time window will
# be used.
#
VALID_BEG =
```

(continues on next page)

(continued from previous page)

```
VALID_END =

##
#
# MET TC-Pairs
#
##

#
# Run MET tc_pairs by indicating the top-level directories for the A-deck
# and B-deck files. Set to 'yes' to run using top-level directories, 'no'
# if you want to run tc_pairs on files paired by the wrapper.
TC_PAIRS_READ_ALL_FILES = no

# List of models to be used (white space or comma separated) eg: DSHP, LGEM, HWRF
# If no models are listed, then process all models in the input file(s).
MODEL =

# List of storm ids of interest (space or comma separated) e.g.: AL112012, AL122012
# If no storm ids are listed, then process all storm ids in the input file(s).
TC_PAIRS_STORM_ID =

# Basins (of origin/region).  Indicate with space or comma-separated list of regions, eg.
↪AL: for North Atlantic,
# WP: Western North Pacific, CP: Central North Pacific, SH: Southern Hemisphere, IO: North
↪Indian Ocean, LS: Southern
# Hemisphere
TC_PAIRS_BASIN =

# Cyclone, a space or comma-separated list of cyclone numbers. If left empty, all cyclones
↪will be used.
TC_PAIRS_CYCLONE =

# Storm name, a space or comma-separated list of storm names to evaluate.  If left empty,
↪all storms will be used.
TC_PAIRS_STORM_NAME =

# DLAND file, the full path of the file that contains the gridded representation of the
# minimum distance from land.
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
```

(continues on next page)

(continued from previous page)

```
TC_PAIRS_MISSING_VAL = -9999


#  Plot_TCMPR options, if left unset, default values that are
#  pre-defined in the R utility (packaged with MET) will be used.
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 =
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


# OVERWRITE OPTIONS
# Don't overwrite filter files if they already exist.
# Set to no if you do NOT want to override existing files
```

(continues on next page)

(continued from previous page)

```
# Set to yes if you do want to override existing files

# overwrite modified track data (non-ATCF to ATCF format)
TC_PAIRS_SKIP_IF_REFORMAT_EXISTS = yes

# overwrite tc_pairs output
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = yes

#  FILENAME TEMPLATES
#
[filename_templates]
# Define the format of the filenames

TC_PAIRS_ADECK_TEMPLATE = {date?fmt=%Y%m}/a{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}
TC_PAIRS_BDECK_TEMPLATE = {date?fmt=%Y%m}/b{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}
TC_PAIRS_OUTPUT_TEMPLATE = {date?fmt=%Y%m}/{basin?fmt=%s}q{date?fmt=%Y%m%d%H}.gfso.{cyclone?
↪fmt=%s}

#
#  DIRECTORIES
#
[dir]

CONFIG_DIR = {PARM_BASE}/use_cases/model_applications/tc_and_extra_tc/Plotter_fcstGFS_obsGFS_
↪RPlotting

# track data, set to your data source
TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/model_applications/medium_range/track_data
TC_PAIRS_BDECK_INPUT_DIR = {TC_PAIRS_ADECK_INPUT_DIR}
TC_PAIRS_REFORMAT_DIR = {OUTPUT_BASE}/track_data_atcf
TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs

TCMPR_PLOTTER_TCMPR_DATA_DIR = {TC_PAIRS_OUTPUT_DIR}
TCMPR_PLOTTER_PLOT_OUTPUT_DIR = {OUTPUT_BASE}/tcmpr_plots
```

5.2.10.2.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *TCPairs MET Configuration* (page 243) 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 = TRUE;

//
// 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;
}

//
// 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}
```

5.2.10.2.7 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 Plot-

ter_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.

5.2.10.2.8 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

5.2.10.2.9 Keywords

Note:

- TCPairsToolUseCase
- TCandExtraTCAppUseCase
- FeatureRelativeUseCase
- MediumRangeAppUseCase
- SBUOrgUseCase
- DTCOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.10.3 CyclonePlotter: Extra-TC Tracker and Plotting Capabilities

model_applications/tc_and_extra_tc/CyclonePlotter_fcstGFS_obsGFS_UserScript_ExtraTC.conf

5.2.10.3.1 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)

5.2.10.3.2 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 1525) section for more information.

Data Source: GFS

5.2.10.3.3 External Dependencies

You will need to use a version of Python 3.6+ that has the following packages installed:

- cartopy
- matplotlib

5.2.10.3.4 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).

5.2.10.3.5 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.

5.2.10.3.6 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

```
#
#  CONFIGURATION
#
[config]

# Looping by times: steps through each 'task' in the PROCESS_LIST for each
# defined time, and repeats until all times have been evaluated.
LOOP_ORDER = processes
```

(continues on next page)

(continued from previous page)

```
# Configuration files
TC_PAIRS_CONFIG_FILE = {PARM_BASE}/met_config/TCPairsConfig_wrapped

# 'Tasks' to be run
PROCESS_LIST = UserScript, TCPairs, CyclonePlotter

LOOP_BY = INIT


# The init time begin and end times, increment, and last init hour.
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2020100700
INIT_END = 2020100700

# This is the step-size. Increment in seconds from the begin time to the end time
# set to 6 hours = 21600 seconds
INIT_INCREMENT = 21600

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID

USER_SCRIPT_PATH = {PARM_BASE}/use_cases/model_applications/tc_and_extra_tc/CyclonePlotter_
↪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}

# A list of times to include, in format YYYYMMDD_hh
TC_PAIRS_INIT_INCLUDE =

# A list of times to exclude, in format YYYYMMDD_hh
TC_PAIRS_INIT_EXCLUDE =

# Specify model init time window in format YYYYMM[DD[_hh]]
# Only tracks that fall within the initialization time window will be used
TC_PAIRS_INIT_BEG =
TC_PAIRS_INIT_END =

# Specify model valid time window in format YYYYMM[DD[_hh]]
# Only tracks that fall within the valid time window will be used
TC_PAIRS_VALID_BEG =
TC_PAIRS_VALID_END =
```

(continues on next page)

(continued from previous page)

```
#
# Run MET tc_pairs by indicating the top-level directories for the A-deck and B-deck files.␣
↪Set to 'yes' to
# run using top-level directories, 'no' if you want to run tc_pairs on files paired by the␣
↪wrapper.
TC_PAIRS_READ_ALL_FILES = no

# set to true or yes to reformat track data into ATCF format expected by tc_pairs
TC_PAIRS_REFORMAT_DECK = no

# OVERWRITE OPTIONS
# Don't overwrite filter files if they already exist.
# Set to yes if you do NOT want to override existing files
# Set to no if you do want to override existing files
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = no

#
# MET TC-Pairs
#
# List of models to be used (white space or comma separated) eg: DSHP, LGEM, HWRF
# If no models are listed, then process all models in the input file(s).
MODEL =

#TC_PAIRS_DESC =

# List of storm ids of interest (space or comma separated) e.g.: AL112012, AL122012
# If no storm ids are listed, then process all storm ids in the input file(s).
TC_PAIRS_STORM_ID =

# Basins (of origin/region).  Indicate with space or comma-separated list of regions, eg.␣
↪AL: for North Atlantic,
# WP: Western North Pacific, CP: Central North Pacific, SH: Southern Hemisphere, IO: North␣
↪Indian Ocean, LS: Southern
# Hemisphere
TC_PAIRS_BASIN =

# Cyclone, a space or comma-separated list of cyclone numbers. If left empty, all cyclones␣
↪will be used.
TC_PAIRS_CYCLONE =

# Storm name, a space or comma-separated list of storm names to evaluate.  If left empty,␣
↪all storms will be used.
TC_PAIRS_STORM_NAME =
```

(continues on next page)

(continued from previous page)

```
# DLAND file, the full path of the file that contains the gridded representation of the
# minimum distance from land.
TC_PAIRS_DLAND_FILE = MET_BASE/tc_data/dland_global_tenth_degree.nc

# setting this so that when verifying against analysis track, the union of points are written
TC_PAIRS_MET_CONFIG_OVERRIDES = match_points = FALSE;

##
# only 00, 06, 12, and 18z init times are supported in NOAA website,
# so for consistency, these are the only options for METplus.
#
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

##
# Indicate the region (i.e. define a bounding box) to plot
#

# Set to Y[y]es or True to plot entire global extent, N[n]o or False
# to generate a plot of a defined region of the world (and define lons and
# lats below).
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

##
#  Indicate the size of symbol (point size)
CYCLONE_PLOTTER_CIRCLE_MARKER_SIZE = 4
CYCLONE_PLOTTER_CROSS_MARKER_SIZE = 6

##
# Indicate text size of annotation label
```

(continues on next page)

(continued from previous page)

```
CYCLONE_PLOTTER_ANNOTATION_FONT_SIZE = 3

# Indicate the text size for the legend labels
CYCLONE_PLOTTER_LEGEND_FONT_SIZE = 3

##
# Resolution of saved plot in dpi (dots per inch)
# Set to 0 to allow Matplotlib to determine, based on your computer
CYCLONE_PLOTTER_RESOLUTION_DPI = 400

CYCLONE_PLOTTER_GENERATE_TRACK_ASCII = yes

CYCLONE_PLOTTER_ADD_WATERMARK = False

#
#  DIRECTORIES
#
[dir]
# Location of input track data directory
# for ADECK and BDECK data

USER_SCRIPT_OUTPUT_DIR = {OUTPUT_BASE}/decks

TC_PAIRS_ADECK_INPUT_DIR = {USER_SCRIPT_OUTPUT_DIR}/adeck
TC_PAIRS_BDECK_INPUT_DIR = {USER_SCRIPT_OUTPUT_DIR}/adeck

TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs

CYCLONE_PLOTTER_INPUT_DIR = {TC_PAIRS_OUTPUT_DIR}
CYCLONE_PLOTTER_OUTPUT_DIR = {OUTPUT_BASE}/cyclone

[filename_templates]
TC_PAIRS_ADECK_TEMPLATE = adeck.{init?fmt=%Y%m%d%H}.{cyclone}.dat
TC_PAIRS_BDECK_TEMPLATE = adeck.{init?fmt=%Y%m%d%H}.{cyclone}.dat
TC_PAIRS_OUTPUT_TEMPLATE = tc_pairs.{init?fmt=%Y%m%d%H}.{cyclone}
```

5.2.10.3.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *TCPairs MET Configuration* (page 243) 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 = TRUE;

//
// 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;
}

//
// 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}
```

5.2.10.3.8 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
```

(continues on next page)

(continued from previous page)

```
#    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

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}.'
```

(continues on next page)

(continued from previous page)

```
# 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]

# 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
```

(continues on next page)

(continued from previous page)

```
    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]

    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")
```

5.2.10.3.9 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.

5.2.10.3.10 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.

5.2.10.3.11 Keywords

Note:

- TCPairsToolUseCase
- SBUOrgUseCase
- CyclonePlotterUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.10.4 Grid-Stat: Verification of TC forecasts against merged TDR data

model_applications/tc_and_extra_tc/GridStat_fcstHAFS_obsTDR _NetCDF.conf

5.2.10.4.1 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.

5.2.10.4.2 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

5.2.10.4.3 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.

5.2.10.4.4 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,*,*)")

5.2.10.4.5 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

```
# GridStat METplus Configuration

# section heading for [config] variables - all items below this line and
# before the next section heading correspond to the [config] section
[config]

# List of applications to run - only GridStat for this case
PROCESS_LIST = GridStat

# time looping - options are INIT, VALID, RETRO, and REALTIME
# If set to INIT or RETRO:
```

(continues on next page)

(continued from previous page)

```
#   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
LOOP_BY = VALID

# Format of INIT_BEG and INT_END using % items
# %Y = 4 digit year, %m = 2 digit month, %d = 2 digit day, etc.
# see www.strftime.org for more information
# %Y%m%d%H expands to YYYYMMDDHH
VALID_TIME_FMT = %Y%m%d%H

# Start time for METplus run - must match INIT_TIME_FMT
VALID_BEG = 2019082912

# End time for METplus run - must match INIT_TIME_FMT
VALID_END = 2019082912

# Increment between METplus runs (in seconds if no units are specified)
#  Must be >= 60 seconds
VALID_INCREMENT = 21600

# List of forecast leads to process for each run time (init or valid)
# In hours if units are not specified
# If unset, defaults to 0 (don't loop through forecast leads)
LEAD_SEQ = 0,6,12,18

# Order of loops to process data - Options are times, processes
# Not relevant if only one item is in the PROCESS_LIST
# times = run all wrappers in the PROCESS_LIST for a single run time, then
#   increment the run time and run all wrappers again until all times have
#   been evaluated.
# processes = run the first wrapper in the PROCESS_LIST for all times
#   specified, then repeat for the next item in the PROCESS_LIST until all
#   wrappers have been run
LOOP_ORDER = times


# Verbosity of MET output - overrides LOG_VERBOSITY for GridStat only
LOG_GRID_STAT_VERBOSITY = 200

# Location of MET config file to pass to GridStat
# References CONFIG_DIR from the [dir] section
GRID_STAT_CONFIG_FILE = {CONFIG_DIR}/GridStatConfig_wrapped
GRID_STAT_OUTPUT_FLAG_FHO = BOTH
GRID_STAT_OUTPUT_FLAG_CTC = STAT
```

(continues on next page)

(continued from previous page)

```
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 to remap data. Value is set as the 'to_grid' variable in the 'regrid' dictionary
# See MET User's Guide for more information
GRID_STAT_REGRID_TO_GRID = OBS

# Name to identify model (forecast) data in output
MODEL = HAFS

# Name to identify observation data in output
OBTYPE = TDR

# add list of missions separated by commas
CUSTOM_LOOP_LIST = 190829H1

# List of variables to compare in GridStat - FCST_VAR1 variables correspond
#  to OBS_VAR1 variables
# Note [FCST/OBS/BOTH]_GRID_STAT_VAR<n>_NAME can be used instead if different evaluations
# are needed for different tools

FCST_VAR1_NAME =  u
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_VAR<n>_LEVELS dimensions are (valid_time, lev, latitude, longitude)
FCST_VAR1_LEVELS =  "(0,1,*,*)"
FCST_GRID_STAT_INPUT_DATATYPE = NETCDF_NCCF

# 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


# Name of observation variable 1
# In this example the variable is merged_zonal_wind
#
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}
```

(continues on next page)

(continued from previous page)

```
#Thresholds for categorical statistics
FCST_VAR1_THRESH = gt10.0, gt20.0, lt-10.0, lt-20.0
OBS_VAR1_THRESH = gt10.0, gt20.0, lt-10.0, lt-20.0

# Time relative to valid time (in seconds) to allow files to be considered
#  valid. Set both BEGIN and END to 0 to require the exact time in the filename
FCST_GRID_STAT_FILE_WINDOW_BEGIN = 0
FCST_GRID_STAT_FILE_WINDOW_END = 0

# MET GridStat neighborhood values
# See the MET User's Guide GridStat section for more information
# width value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_WIDTH = 1

# shape value passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_SHAPE = SQUARE

# cov thresh list passed to nbrhd dictionary in the MET config file
GRID_STAT_NEIGHBORHOOD_COV_THRESH = >=0.5

# Set to true to run GridStat separately for each field specified
# Set to false to create one run of GridStat per run time that
#   includes all fields specified.
GRID_STAT_ONCE_PER_FIELD = False

# Set to true if forecast data is probabilistic
FCST_IS_PROB = false

# Only used if FCST_IS_PROB is true - sets probabilistic threshold
FCST_GRID_STAT_PROB_THRESH = ==0.1

# Set to true if observation data is probabilistic
#  Only used if configuring forecast data as the 'OBS' input
OBS_IS_PROB = false

# Only used if OBS_IS_PROB is true - sets probabilistic threshold
OBS_GRID_STAT_PROB_THRESH = ==0.1

GRID_STAT_OUTPUT_PREFIX = {MODEL}_vs_{OBTYPE}

# End of [config] section and start of [dir] section
[dir]

# location of configuration files used by MET applications
CONFIG_DIR={PARM_BASE}/met_config
```

(continues on next page)

(continued from previous page)

```
# directory containing forecast input to GridStat
FCST_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/GridStat_fcstHAFS_
↪obsTDR_NetCDF/hafs_height

# directory containing observation input to GridStat
OBS_GRID_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/GridStat_fcstHAFS_
↪obsTDR_NetCDF/obs

# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_DIR =

# directory containing climatology mean input to GridStat
# Not used in this example
GRID_STAT_CLIMO_STDEV_INPUT_DIR =

# directory to write output from GridStat
GRID_STAT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/tc_and_extra_tc/tdr

# End of [dir] section and start of [filename_templates] section
[filename_templates]

# Template to look for forecast input to GridStat relative to FCST_GRID_STAT_INPUT_DIR
FCST_GRID_STAT_INPUT_TEMPLATE = dorian05l.{init?fmt=%Y%m%d%H}.hafsprs.synoptic.0p03.f{lead?
↪fmt=%HHH}.nc4

# Template to look for observation input to GridStat relative to OBS_GRID_STAT_INPUT_DIR
OBS_GRID_STAT_INPUT_TEMPLATE = PYTHON_NUMPY


# Optional subdirectories relative to GRID_STAT_OUTPUT_DIR to write output from GridStat
GRID_STAT_OUTPUT_TEMPLATE = {init?fmt=%Y%m%d%H}

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_MEAN_INPUT_
↪DIR
# Not used in this example
GRID_STAT_CLIMO_MEAN_INPUT_TEMPLATE =

# Template to look for climatology input to GridStat relative to GRID_STAT_CLIMO_STDEV_INPUT_
↪DIR
# Not used in this example
GRID_STAT_CLIMO_STDEV_INPUT_TEMPLATE =

# Used to specify one or more verification mask files for GridStat
```

(continues on next page)

(continued from previous page)

```
# Not used for this example
GRID_STAT_VERIFICATION_MASK_TEMPLATE =
```

5.2.10.4.6 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}

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

//
// 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}

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

//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.

5.2.10.4.7 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

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
```

(continues on next page)

(continued from previous page)

```
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

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):
  ###########################################
```

(continues on next page)

(continued from previous page)

```
##
##  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

    # 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))
```

(continues on next page)

(continued from previous page)

```
    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")),

      '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),
      }
```

(continues on next page)

(continued from previous page)

```
    }

    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)
```

5.2.10.4.8 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.

5.2.10.4.9 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.

5.2.10.4.10 Keywords

Note:

- TCandExtraTCAppUseCase
- GridStatToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.10.5 Point-Stat: Standard Verification for CONUS Surface

model_applications/tc_and_extra_tc/UserScript_ASCII2NC_PointStat_fcstHAFS_obsFRD _NetCDF.conf

5.2.10.5.1 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.

5.2.10.5.2 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 1555) section for more information.

Dropsonde Data Source: Hurricane Research Division Sonde Archive

5.2.10.5.3 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.

5.2.10.5.4 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

5.2.10.5.5 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]

## Configuration-related settings such as the process list, begin and end times, etc.
PROCESS_LIST = UserScript(untar_drop_file), Ascii2nc, PointStat

USER_SCRIPT_RUNTIME_FREQ = RUN_ONCE_PER_INIT_OR_VALID
USER_SCRIPT_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/dropsonde/obs
```

(continues on next page)

(continued from previous page)

```
USER_SCRIPT_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/tc_and_extra_tc/dropsonde/obs
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_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

## LOOP_ORDER
## Options are: processes, times
## Looping by time- runs all items in the PROCESS_LIST for each
## initialization time and repeats until all times have been evaluated.
## Looping by processes- run each item in the PROCESS_LIST for all
## specified initialization times then repeat for the next item in the
## PROCESS_LIST.
LOOP_ORDER = processes

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


# Logging levels: DEBUG, INFO, WARN, ERROR (most verbose is DEBUG)
LOG_LEVEL = DEBUG

## MET Configuration files for point_stat

# Message types, if all message types are to be returned, leave this empty,
# otherwise indicate the message types of interest.
POINT_STAT_MESSAGE_TYPE = ADPUPA
POINT_STAT_STATION_ID =
```

(continues on next page)

(continued from previous page)

```
# Verification Masking regions
# Indicate which grid and polygon masking region, if applicable
POINT_STAT_GRID = FULL

# List of full path to poly masking files.  NOTE: Only short lists of poly
# files work (those that fit on one line), a long list will result in an
# environment variable that is too long, resulting in an error.  For long
# lists of poly masking files (i.e. all the mask files in the NCEP_mask
# directory), define these in the MET point_stat configuration file.
POINT_STAT_POLY =

# For both pb2nc and point_stat, the obs_window dictionary:
OBS_WINDOW_BEGIN = -5400
OBS_WINDOW_END = 5400

# Model/fcst and obs name, e.g. GFS, NAM, GDAS, etc.
MODEL = HAFS
OBTYPE = drop

# Variables and levels as specified in the field dictionary of the MET
# point_stat configuration file. Specify as FCST_VARn_NAME, FCST_VARn_LEVELS,
# (optional) FCST_VARn_OPTION

BOTH_VAR1_NAME = TMP
BOTH_VAR1_LEVELS = P925-950, P850-800, P700-650

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
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

# Regrid to specified grid.  Indicate NONE if no regridding, or the grid id
# (e.g. G212)
POINT_STAT_REGRID_TO_GRID = NONE
```

(continues on next page)

(continued from previous page)

```
LOG_POINT_STAT_VERBOSITY=5

[dir]
TAR_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/dropsonde/obs
FCST_POINT_STAT_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/dropsonde
OBS_POINT_STAT_INPUT_DIR = {OUTPUT_BASE}/model_applications/tc_and_extra_tc/dropsonde/
↪ascii2nc
ASCII2NC_OUTPUT_DIR = {OUTPUT_BASE}/model_applications/tc_and_extra_tc/dropsonde/ascii2nc
POINT_STAT_OUTPUT_DIR = {OUTPUT_BASE}/{OBTYPE}

[filename_templates]

USER_SCRIPT_ARGUMENTS = {USER_SCRIPT_INPUT_DIR} {valid?fmt=%Y%m%d} {USER_SCRIPT_OUTPUT_DIR}
ASCII2NC_OUTPUT_TEMPLATE = drop{valid?fmt=%Y%m%d}.nc
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_TEMPLATE = {ASCII2NC_OUTPUT_TEMPLATE}
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}"
```

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.

5.2.10.5.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Ascii2NcConfig_wrapped

Note: See the *ASCII2NC MET Configuration* (page 80) 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}
```

PointStatConfig_wrapped

Note: See the *PointStat MET Configuration* (page 194) 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 ];
```

(continues on next page)

(continued from previous page)

```
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 =
${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_GRID}
   ${METPLUS_MASK_POLY}
   ${METPLUS_MASK_SID}
   //llpnt =
   ${METPLUS_MASK_LLPNT}
}

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

//
// Confidence interval settings
//
ci_alpha  = [ 0.05 ];
```

(continues on next page)

(continued from previous page)

```
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}

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

tmp_dir = "${MET_TMP_DIR}";

// output_prefix =
${METPLUS_OUTPUT_PREFIX}
//version        = "V10.0.0";

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

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.10.5.7 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/hrd

```
#! /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/hrc

```
########################################################################
#
# 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)))
```

5.2.10.5.8 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.

5.2.10.5.9 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

5.2.10.5.10 Keywords

Note:

- TCandExtraTCAppUseCase
- UserScriptUseCase
- PointStatToolUseCase
- ASCII2NCToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.10.6 TCGen: 2021 Global Forecast System (GFS) Tropical Cyclone Genesis Forecast

model_applications/tc_and_extra_tc/TCGen_fcstGFS_obsBDECK_2021season.conf

5.2.10.6.1 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.

5.2.10.6.2 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 1569) section for more information.

Data Source: NHC ftp.noaa.gov/atcf
Data Source: www.nhc.noaa.gov/archive/wgtwo/

5.2.10.6.3 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.

5.2.10.6.4 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

5.2.10.6.5 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]

# Looping by times: steps through each 'task' in the PROCESS_LIST for each
# defined time, and repeats until all times have been evaluated.
LOOP_ORDER = times

# 'Tasks' to be run
PROCESS_LIST = TCGen

LOOP_BY = INIT

# The init time
INIT_TIME_FMT = %Y
INIT_BEG = 2021


LOG_TC_GEN_VERBOSITY = 2

# optional list of strings to loop over and call the tool multiple times
# value of each item can be referenced in filename templates with {custom?fmt=%s}
TC_GEN_CUSTOM_LOOP_LIST =


# I/O Configurations

# Location of input data directory for track data
TC_GEN_TRACK_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/TCGen_fcstGFS_
↪obsBDECK_2021season/abdeck/
TC_GEN_TRACK_INPUT_TEMPLATE = *.dat

# Location of input data directory for genesis data
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
```

(continues on next page)

(continued from previous page)

```
TC_GEN_SHAPE_INPUT_TEMPLATE = */{init?fmt=%Y}*/gtwo*.shp


# directory to write output files generated by tc_gen
TC_GEN_OUTPUT_DIR = {OUTPUT_BASE}/model_application/tc_and_extra_tc/TCGen
TC_GEN_OUTPUT_TEMPLATE = tc_gen


# MET Configurations

TC_GEN_CONFIG_FILE = {PARM_BASE}/met_config/TCGenConfig_wrapped

# The following variables set values in the MET configuration file used by this example
# Leaving these values commented will use the value found in the default MET configuration␣
↪file
# See the MET documentation for this tool for more information on the settings

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 = GFS

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 =

# sets METPLUS_LEAD in the wrapped MET config file
LEAD_SEQ =

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 = -6
TC_GEN_GENESIS_MATCH_WINDOW_END = 6
```

(continues on next page)

(continued from previous page)

```
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

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
```

5.2.10.6.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *TC-Gen MET Configuration* (page 227) 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}
```

5.2.10.6.7 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.

5.2.10.6.8 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

5.2.10.6.9 Keywords

Note:

- TCGenToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.10.7 Cyclone Plotter: From TC-Pairs Output

model_applications/tc_and_extra_tc/Plotter_fcstGFS _obsGFS_ExtraTC.conf

5.2.10.7.1 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.

5.2.10.7.2 Datasets

- Forecast dataset: ADeck modified-ATCF tropical cyclone data
- Observation dataset: BDeck modified-ATCF "best-track" tropical cyclone data

5.2.10.7.3 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.

5.2.10.7.4 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.

5.2.10.7.5 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

```
[dir]
## Dirs below used by tc_pairs_wrapper module.
# --------------------------------------------
# track data, set to your data source

TC_PAIRS_ADECK_INPUT_DIR = {INPUT_BASE}/model_applications/tc_and_extra_tc/track_data
TC_PAIRS_BDECK_INPUT_DIR = {TC_PAIRS_ADECK_INPUT_DIR}

# Where modified track files are saved
TC_PAIRS_REFORMAT_DIR = {OUTPUT_BASE}/track_data_atcf
TC_PAIRS_OUTPUT_DIR = {OUTPUT_BASE}/tc_pairs

## Dirs below used by cyclone_plotter_wrapper module.
# --------------------------------------------------
CYCLONE_PLOTTER_INPUT_DIR = {OUTPUT_BASE}/tc_pairs
CYCLONE_PLOTTER_OUTPUT_DIR = {OUTPUT_BASE}/cyclone

[filename_templates]
TC_PAIRS_ADECK_TEMPLATE = {date?fmt=%Y%m}/a{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}
TC_PAIRS_BDECK_TEMPLATE = {date?fmt=%Y%m}/b{basin?fmt=%s}q{date?fmt=%Y%m}*.gfso.{cyclone?fmt=
↪%s}
TC_PAIRS_OUTPUT_TEMPLATE = {date?fmt=%Y%m}/{basin?fmt=%s}q{date?fmt=%Y%m%d%H}.gfso.{cyclone?
↪fmt=%s}

[config]
# ================================================
##
# EXTRA TROPICAL CYCLONE PLOT OPTIONS...
#
PROCESS_LIST = TCPairs, CyclonePlotter
LOOP_ORDER = processes
```

(continues on next page)

(continued from previous page)

```
LOOP_BY = init

## Config options below used by tc_pairs_wrapper module.
# -------------------------------------------------------
##
#
# MET TC-Pairs
#
#
##

#
# Generate the tc-pairs data of interest
#
# Configuration files
TC_PAIRS_CONFIG_FILE = {PARM_BASE}/met_config/TCPairsConfig_wrapped

INIT_TIME_FMT = %Y%m%d
INIT_BEG = 20150301
INIT_END = 20150330
INIT_INCREMENT = 21600     ;; 6 hours


TC_PAIRS_RUN_ONCE = True

# A list of times to include, in format YYYYMMDD_hh
INIT_INCLUDE =

# A list of times to exclude, in format YYYYMMDD_hh
INIT_EXCLUDE =

#
# Specify model valid time window in format YYYYMM[DD[_hh]].
# Only tracks that fall within the valid time window will
# be used.
VALID_BEG =
VALID_END =

#
# Run MET tc_pairs by indicating the top-level directories for the A-deck and B-deck files.␣
↪Set to 'yes' to
# run using top-level directories, 'no' if you want to run tc_pairs on files paired by the␣
↪wrapper.
TC_PAIRS_READ_ALL_FILES = no
```

(continues on next page)

(continued from previous page)

```
#
# MET TC-Pairs
#
# List of models to be used (white space or comma separated) eg: DSHP, LGEM, HWRF
# If no models are listed, then process all models in the input file(s).
MODEL =

# List of storm ids of interest (space or comma separated) e.g.: AL112012, AL122012
# If no storm ids are listed, then process all storm ids in the input file(s).
TC_PAIRS_STORM_ID =

# Basins (of origin/region).  Indicate with space or comma-separated list of regions, eg.␣
↪AL: for North Atlantic,
# WP: Western North Pacific, CP: Central North Pacific, SH: Southern Hemisphere, IO: North␣
↪Indian Ocean, LS: Southern
# Hemisphere
TC_PAIRS_BASIN =

# Cyclone, a space or comma-separated list of cyclone numbers. If left empty, all cyclones␣
↪will be used.
TC_PAIRS_CYCLONE =

# Storm name, a space or comma-separated list of storm names to evaluate.  If left empty,␣
↪all storms will be used.
TC_PAIRS_STORM_NAME =


# DLAND file, the full path of the file that contains the gridded representation of the
# minimum distance from land.
# Setting this causes tc_pairs to run approx 4x slower
TC_PAIRS_DLAND_FILE = {MET_INSTALL_DIR}/share/met/tc_data/dland_global_tenth_degree.nc

## tc-pairs filtering options
TC_PAIRS_REFORMAT_DECK = yes
TC_PAIRS_REFORMAT_TYPE = SBU

TC_PAIRS_MISSING_VAL_TO_REPLACE = -99
TC_PAIRS_MISSING_VAL = -9999

# SKIP OPTIONS
# Skip processing files if the output already exists.
# Set to yes if you do NOT want to override existing files
# Set to no if you do want to override existing files
TC_PAIRS_SKIP_IF_REFORMAT_EXISTS = yes
TC_PAIRS_SKIP_IF_OUTPUT_EXISTS = yes
```

(continues on next page)

(continued from previous page)

```
##
# CYCLONE PLOTTER
#
## Config options below used by cyclone_plotter_wrapper module.
# --------------------------------------------------------------
##

#
# Specify the YMD of tracks of interest
#
CYCLONE_PLOTTER_INIT_DATE = 20150301

##
# only 00, 06, 12, and 18z init times are supported in NOAA website,
# so for consistency, these are the only options for METplus.
#
CYCLONE_PLOTTER_INIT_HR = 12 ;; hh format
CYCLONE_PLOTTER_MODEL = GFSO
CYCLONE_PLOTTER_PLOT_TITLE = Model Forecast Storm Tracks

##
# Indicate the region (i.e. define a bounding box) to plot
#

# Set to Y[y]es or True to plot entire global extent, N[n]o or False
# to generate a plot of a defined region of the world (and define lons and
# lats below).
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

#  Indicate the size of symbol (point size)
CYCLONE_PLOTTER_CIRCLE_MARKER_SIZE = 2
```

(continues on next page)

(continued from previous page)

```
CYCLONE_PLOTTER_CROSS_MARKER_SIZE = 3

##
# Indicate text size of annotation label
CYCLONE_PLOTTER_ANNOTATION_FONT_SIZE = 3

# Indicate the text size for the legend labels
CYCLONE_PLOTTER_LEGEND_FONT_SIZE = 3

##
#  Turn on/off the generation of an ASCII output file listing all the
#  tracks that are in the plot.  This can be helpful in debugging or verifying
#  that what is plotted is consistent with the data.
#
CYCLONE_PLOTTER_GENERATE_TRACK_ASCII = yes

CYCLONE_PLOTTER_ADD_WATERMARK = False

##
# Resolution of saved plot in dpi (dots per inch)
# Set to 0 to allow Matplotlib to determine, based on your computer
CYCLONE_PLOTTER_RESOLUTION_DPI = 400
```

5.2.10.7.6 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *TCPairs MET Configuration* (page 243) section of the User's Guide for more information on the environment variables used in the file below:

```
////////////////////////////////////////////////////////////////////////////////
//
// Default TCPairs configuration file
//
////////////////////////////////////////////////////////////////////////////////

//
```

(continues on next page)

(continued from previous page)

```
// 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}
// init_inc =
${METPLUS_INIT_INC}
// init_exc =
${METPLUS_INIT_EXC}

// valid_inc =
```

(continues on next page)

(continued from previous page)

```
${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.
//
//interp12 =
${METPLUS_INTERP12}

//
// Specify how consensus forecasts should be defined
```

(continues on next page)

(continued from previous page)

```
//
//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 = TRUE;

//
// 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;
```

(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 = "V9.0";

tmp_dir = "${MET_TMP_DIR}";

${METPLUS_MET_CONFIG_OVERRIDES}
```

5.2.10.7.7 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.

5.2.10.7.8 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

5.2.10.7.9 Keywords

Note:

- TCPairsToolUseCase
- CyclonePlotterUseCase
- FeatureRelativeUseCase
- MediumRangeAppUseCase
- NOAAEMCOrgUseCase
- SBUOrgUseCase
- DTCOrgUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) 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)

5.2.10.8 CycloneVerification: TC Verification Compare ADECK vs BDECK

model_applications/tc_and_extra_tc/TCPairs_TCStat_fcstADECK_obsBDECK_ATCF_BasicExample.conf

5.2.10.8.1 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.

5.2.10.8.2 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 1592) section for more information.

Data Source: NHC ftp.noaa.gov/atcf

5.2.10.8.3 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).

5.2.10.8.4 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.

5.2.10.8.5 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.).

5.2.10.8.6 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

```
#
#  CONFIGURATION
#
[config]

# Looping by times: steps through each 'task' in the PROCESS_LIST for each
# defined time, and repeats until all times have been evaluated.

# 'Tasks' to be run
PROCESS_LIST = TCPairs, TCStat

LOOP_BY = INIT


# The init time begin and end times, increment, and last init hour.
INIT_TIME_FMT = %Y%m%d%H
INIT_BEG = 2021082500
INIT_END = 2021083000
```

(continues on next page)

(continued from previous page)

```
# This is the step-size. Increment in seconds from the begin time to the end time
# set to 6 hours = 21600 seconds
INIT_INCREMENT = 21600

#
# Run MET tc_pairs by indicating the top-level directories for the A-deck and B-deck files.␣
↪Set to 'yes' to
# run using top-level directories, 'no' if you want to run tc_pairs on files paired by the␣
↪wrapper.
TC_PAIRS_READ_ALL_FILES = no


#
# MET TC-Pairs
#
# List of models to be used (white space or comma separated) eg: DSHP, LGEM, HWRF
# If no models are listed, then process all models in the input file(s).
MODEL = OFCL, HWRF

# List of storm ids of interest (space or comma separated) e.g.: AL112012, AL122012
# If no storm ids are listed, then process all storm ids in the input file(s).
#TC_PAIRS_STORM_ID = AL092021, AL102021

# Basins (of origin/region).  Indicate with space or comma-separated list of regions, eg.␣
↪AL: for North Atlantic,
# WP: Western North Pacific, CP: Central North Pacific, SH: Southern Hemisphere, IO: North␣
↪Indian Ocean, LS: Southern
# Hemisphere
TC_PAIRS_BASIN = AL

# Cyclone, a space or comma-separated list of cyclone numbers. If left empty, all cyclones␣
↪will be used.
TC_PAIRS_CYCLONE = 09, 10

# DLAND file, the full path of the file that contains the gridded representation of the
# minimum distance from land.
TC_PAIRS_DLAND_FILE = MET_BASE/tc_data/dland_global_tenth_degree.nc

# setting this so that when verifying against analysis track, the union of points are written
TC_PAIRS_MET_CONFIG_OVERRIDES = match_points = TRUE;

#
# MET TC-Stat
#
```

(continues on next page)

(continued from previous page)

```
#The line_type field stratifies by the line_type column.
TC_STAT_LINE_TYPE = TCMPR

#The column_str_name and column_str_val fields stratify by performing string matching on non-
↪numeric data columns.
TC_STAT_COLUMN_STRING_NAME = LEVEL
TC_STAT_COLUMN_STRING_VAL = HU,SD,SS,TS,TD

#The water_only flag stratifies by only using points where both the amodel and bmodel tracks␣
↪are over water.
TC_STAT_WATER_ONLY = FALSE

#
#  DIRECTORIES
#
# Location of input track data directory
# for ADECK and BDECK data

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}

TC_STAT_LOOKIN_DIR = {TC_PAIRS_OUTPUT_DIR}
TC_STAT_OUTPUT_DIR = {OUTPUT_BASE}/tc_stat

TC_STAT_JOB_ARGS = -job filter -dump_row {TC_STAT_OUTPUT_DIR}/tc_stat_summary.tcst
```

5.2.10.8.7 MET Configuration

METplus sets environment variables based on user settings in the METplus configuration file. See *How METplus controls MET config file settings* (page 54) 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 68)

Note: See the *TCPairs MET Configuration* (page 243) 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 = TRUE;

//
// 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;
}

//
// 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.
//

//
// Stratify by the AMODEL or BMODEL columns.
//
${METPLUS_AMODEL}
${METPLUS_BMODEL}
```

(continues on next page)

(continued from previous page)

```
//
// 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_INCLUDE}
${METPLUS_INIT_EXCLUDE}

//
// Stratify by the VALID times.
//
${METPLUS_VALID_BEG}
${METPLUS_VALID_END}
${METPLUS_VALID_INCLUDE}
```

(continues on next page)

(continued from previous page)

```
${METPLUS_VALID_EXCLUDE}

//
// 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 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}
```

(continues on next page)

(continued from previous page)

```
//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}

//
// 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}
```

(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}
```

5.2.10.8.8 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.

5.2.10.8.9 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

5.2.10.8.10 Keywords

Note:

- TCPairsToolUseCase
- TCStatToolUseCase

Navigate to the *METplus Quick Search for Use Cases* (page 1595) page to discover other similar use cases.

Total running time of the script: (0 minutes 0.000 seconds)

Chapter 6

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.

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*
PointStat: *PointStatToolUseCase*
RegridDataPlane: *RegridDataPlaneToolUseCase*
SeriesAnalysis: *SeriesAnalysisUseCase*
StatAnalysis: *StatAnalysisToolUseCase*
TCMPRPlotter: *TCMPRPlotterUseCase*
TCGen: *TCGenToolUseCase*
TCPairs: *TCPairsToolUseCase*
TCRMW: *TCRMWToolUseCase*

TCStat: *TCStatToolUseCase*

6.1 Use Cases by Application:

Air Quality and Composition: *AirQualityAndCompAppUseCase*
Climate: *ClimateAppUseCase*
Convection Allowing Models: *ConvectionAllowingModelsAppUseCase*
Data Assimilation: *DataAssimilationAppUseCase*
Ensemble: *EnsembleAppUseCase*
Marine and Cryosphere: *MarineAndCryosphereAppUseCase*
Medium Range: *MediumRangeAppUseCase*
Precipitation: *PrecipitationAppUseCase*
Space Weather: *SpaceWeatherAppUseCase*
Subseasonal to Seasonal: *S2SAppUseCase*
Tropical Cyclone and Extra-Tropical Cyclone: *TCandExtraTCAppUseCase*

6.2 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*

6.3 Use Cases by METplus Feature:

Introductory Example: *ExampleToolUseCase*
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*

METdbLoad: *METdbLoadUseCase*
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*
Validation of Models or Analyses: *ValidationUseCase*
User Defined Script: *UserScriptUseCase*

6.4 Use cases by File Format:

GEMPAK: *GEMPAKFileUseCase*
GRIB: *GRIBFileUseCase*
GRIB2: *GRIB2FileUseCase*
NetCDF: *NetCDFFileUseCase*
Python Embedding: *PythonEmbeddingFileUseCase*
prepBUFR: *prepBUFRFileUseCase*

Chapter 7

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.

Used by: MakePlots, 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 68) 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 40) 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 40) 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 40) 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 40) for more information.

Used by: GridStat, EnsembleStat, PointStat, MODE, MTD, PCPCombine

CI_METHOD

> **Warning: DEPRECATED:** Please use *MAKE_PLOTS_CI_METHOD*.

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: MakePlots, 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}
```

(continues on next page)

(continued from previous page)

```
[dir]
SERIES_ANALYSIS_OUTPUT_DIR = {OUTPUT_BASE}/{custom?fmt=%s}
```

ith 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.

f 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_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_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_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_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

DATE_TYPE

In StatAnalysis, this specifies the way to treat the date information, where valid options are VALID and INIT.

Used by: MakePlots, 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.

Used by: MakePlots, 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
Set the file_type entry of the ens dictionary in the MET config file for EnsembleStat.

Used by: EnsembleStat

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 40) 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 40) 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 40) 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 40) 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 En-

sembleStat. 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_ENS_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_MEMBER_IDS

Specify the value for 'ens_member_ids' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENS_OBS_THRESH

Sets the ens.obs_thresh value in the ensemble_stat MET config file.

Used by: EnsembleStat

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

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

ENSEMBLE_STAT_ENSEMBLE_FLAG_FREQUENCY

Specify the value for 'ensemble_flag.frequency' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_LATLON

Specify the value for 'ensemble_flag.latlon' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_MAX

Specify the value for 'ensemble_flag.max' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_MEAN

Specify the value for 'ensemble_flag.mean' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_MIN

Specify the value for 'ensemble_flag.min' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_MINUS
Specify the value for 'ensemble_flag.minus' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_NEP
Specify the value for 'ensemble_flag.nep' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_NMEP
Specify the value for 'ensemble_flag.nmep' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_PLUS
Specify the value for 'ensemble_flag.plus' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_RANGE
Specify the value for 'ensemble_flag.range' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_RANK
Specify the value for 'ensemble_flag.rank' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_STDEV
Specify the value for 'ensemble_flag.stdev' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_VLD_COUNT
Specify the value for 'ensemble_flag.vld_count' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_ENSEMBLE_FLAG_WEIGHT
Specify the value for 'ensemble_flag.weight' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

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 68) 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

Specify the value for 'nbrhd_prob.shape' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NBRHD_PROB_VLD_THRESH

Specify the value for 'nbrhd_prob.vld_thresh' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NBRHD_PROB_WIDTH

Specify the value for 'nbrhd_prob.width' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_DX

Specify the value for 'nmep_smooth.gaussian_dx' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NMEP_SMOOTH_GAUSSIAN_RADIUS
Specify the value for 'nmep_smooth.gaussian_radius' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NMEP_SMOOTH_METHOD
Specify the value for 'nmep_smooth.type.method' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NMEP_SMOOTH_SHAPE
Specify the value for 'nmep_smooth.shape' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NMEP_SMOOTH_VLD_THRESH
Specify the value for 'nmep_smooth.vld_thresh' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

ENSEMBLE_STAT_NMEP_SMOOTH_WIDTH
Specify the value for 'nmep_smooth.type.width' in the MET configuration file for EnsembleStat.

Used by: EnsembleStat

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_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_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

EVENT_EQUALIZATION

> **Warning: DEPRECATED:** Please use *MAKE_PLOTS_EVENT_EQUALIZATION*.

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.

Used by: MakePlots, 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

Used by: MakePlots, 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. This is optional in the METplus configuration file for running with *LOOP_ORDER* = times.

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. This is optional in the METplus configuration file for running with *LOOP_ORDER* = times.

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. This is optional in the METplus configuration file for running with *LOOP_ORDER* = times.

Used by: StatAnalysis

FCST_VALID_HOUR_LIST
Specify a list of hours for valid times of forecast files for use in the analysis.

Used by: MakePlots, 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 40) 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 40) 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 40) 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 40) 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. This is optional in the METplus configuration file for running with *LOOP_ORDER* = times.

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_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 47) 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 47) 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_CDF
Specify the value for 'ensemble_flag.climo_cdf' 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_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 68) 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_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 GFDLTracker. 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 68) 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_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 51) 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 68) 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_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_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_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_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: MakePlots, 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 30) 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 30) 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 30) 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 31) 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 30) 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.

Used by: MakePlots, 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.

Used by: MakePlots, 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 68) 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.

JOB_NAME

> **Warning:** **DEPRECATED:** Please use *STAT_ANALYSIS_JOB_NAME* 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 31) 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 31) 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 31) 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. For TCMPRPlotter, this is optional in the METplus configuration file for running with *LOOP_ORDER* = times.

Used by: MakePlots, 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. Everything above this level is sent to standard output. To quiet the output to a comfortable level, set this to "ERROR"

Options (ordered MOST verbose to LEAST verbose): | NOTSET | DEBUG | INFO | WARNING | ERROR | CRITICAL

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 the MET tools is sent to the METplus log file, or individual log files for each MET tool.

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

Control the filename of the METplus log file. Control the timestamp appended to the filename with LOG_TIMESTAMP_TEMPLATE. To turn OFF all logging, do not set this option.

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_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_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_TEMPLATE
Set the timestamp template for the METplus log file. Use Python strftime directives, e.g.%Y%m%d for YYYYMMDD.

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 28) 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: MakePlots, StatAnalysis

LOOP_ORDER
Control the looping order for METplus. Valid options are "times" or "processes". "times" runs all items in the *PROCESS_LIST* for a single run time, then repeat until all times have been evaluated. "processes" runs each item in the *PROCESS_LIST* for all times specified, then repeat for the next item in the *PROCESS_LIST*.

Used by: All

MAKE_PLOTS_AVERAGE_METHOD
The method to use to average the data. Valid options are MEAN, MEDIAN, and AGGREGATION.

Used by: MakePlots

MAKE_PLOTS_CI_METHOD
The method for creating confidence intervals. Valid options are EMC, or NONE.

Used by: MakePlots

MAKE_PLOTS_EVENT_EQUALIZATION

If event equalization is to be used (True) or not (False). If set to True, if any of the listed models are missing data for a particular time, data for all models will be masked out for this time. If set to False, there are no changes to the data.

Used by: MakePlots

MAKE_PLOTS_INPUT_DIR

Directory containing input files used by MakePlots.

Used by: MakePlots

MAKE_PLOTS_OUTPUT_DIR

Directory to write files generated by MakePlots.

Used by: MakePlots

MAKE_PLOTS_SCRIPTS_DIR

Directory to find scripts used by MakePlots.

Used by: MakePlots

MAKE_PLOTS_STATS_LIST

This is a list of the statistics to calculate and create plots for. Specify the list in a comma-separated list, e.g.:

acc, bias, rmse

The list of valid options varies depending on line type that was used during the filtering of stat_analysis_wrapper. For SL1L2, VL1L2 valid options are bias, rms, msess, rsd, rmse_md, rmse_pv, pcor, fbar, and fbar_obar. For SAL1L2, VAL1L2, the valid options is acc. For VCNT, bias, fbar, fbar_obar, speed_err, dir_err, rmsve, vdiff_speed, vdiff_dir, rsd, fbar_speed, fbar_dir, fbar_obar_speed, and fbar_obar_dir. For CTC, rate, baser, frate, orate_frate, baser_frate, accuracy, bias, fbias, pod, hrate, pofd, farate, podn, faratio, csi, ts, gss, ets, hk, tss, pss, hs

Used by: MakePlots

MAKE_PLOTS_VERIF_CASE

Verification case used by MakePlots. Valid options for this include: grid2grid, grid2obs, precip.

Used by: MakePlots

MAKE_PLOTS_VERIF_GRID

Specify a string describing the grid the verification was performed on. This is the name of the grid upon which the verification was done on, ex. G002.

Used by: MakePlots

MAKE_PLOTS_VERIF_TYPE

Specify a string describing the type of verification being performed. For MAKE_PLOTS_VERIF_CASE = grid2grid, valid options are anom, pres, and sfc. For MAKE_PLOTS_VERIF_CASE = grid2obs, valid options are conus_sfc and upper_air. For MAKE_PLOTS_VERIF_CASE = precip, any accumulation amount is valid, ex. A24.

Used by: MakePlots

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/METdatadb 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 51) 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

Provide the absolute path to the METplus final configuration file. This file will contain every configuration option and value used when METplus was run.

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_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 68) for more information

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_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_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: MakePlots, StatAnalysis

MODEL<n>_NAME

> **Warning: DEPRECATED:** Please use *MODEL<n>*.

MODEL<n>_NAME_ON_PLOT

> **Warning: DEPRECATED:** Please use *MODEL<n>_REFERENCE_NAME* instead.

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: MakePlots, StatAnalysis

MODEL<n>_REFERENCE_NAME

Define the name the first model will be listed as on the plots. 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_REFERENCE_NAME
MODEL2_REFERENCE_NAME
...
MODELN_REFERENCE_NAME

Used by: MakePlots, StatAnalysis

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. This is optional in the METplus configuration file for running with *LOOP_ORDER* = times.

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. This is optional in the METplus configuration file for running with *LOOP_ORDER* = times.

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.

Used by: MakePlots, 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 68) 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_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 47) 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 47) 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 47) 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 47) 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 47) 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 47) 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.

Used by: MakePlots, 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

Used by: MakePlots, 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. This is optional in the METplus configuration file for running with *LOOP_ORDER* = times.

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 47) 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 47) 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 47) 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 47) 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 47) 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 47) 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. This is optional in the METplus configuration file for running with *LOOP_ORDER* = times.

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. This is optional in the METplus configuration file for running with *LOOP_ORDER* = times.

Used by: StatAnalysis

OBS_VALID_HOUR_LIST
Specify a list of hours for valid times of observation files for use in the analysis.

Used by: MakePlots, 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 40) 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 40) 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. This is optional in the METplus configuration file for running with *LOOP_ORDER* = times.

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 47) 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 47) 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 68) 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_STATS_LIST

> **Warning: DEPRECATED:** Please use *MAKE_PLOTS_STATS_LIST* instead.

PLOT_TIME

> **Warning: DEPRECATED:** Please use *DATE_TYPE* instead.

PLOT_TYPES

> **Warning: DEPRECATED:** Please use *TCMPR_PLOTTER_PLOT_TYPES* instead.

PLOTTING_OUTPUT_DIR

> **Warning: DEPRECATED:** Please use *MAKE_PLOTS_OUTPUT_DIR* instead.

PLOTTING_SCRIPTS_DIR

> **Warning: DEPRECATED:** Please use *MAKE_PLOTS_SCRIPTS_DIR* 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 regrid dictionary item 'to_grid' in the MET Point2Grid config file. See the MET User's Guide for more information.

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 68) 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_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_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_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 40) 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.

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
Remove staging directory after METplus has completed running if set to True. Set to False to preserve data for subsequent runs.

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_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 68) 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_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 51) 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 uncompress or convert data into for use in METplus.

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_DUMP_ROW_TMPL

> **Warning:** **DEPRECATED:** Please use *MODEL<n>_STAT_ANALYSIS_DUMP_ROW_TEMPLATE* instead.

STAT_ANALYSIS_HSS_EC_VALUE

Specify the value for 'hss_ec_value' in the MET configuration file for StatAnalysis.

Used by: StatAnalysis

STAT_ANALYSIS_JOB_ARGS

Specify stat_analysis job arguments to run. The job arguments that are to be run with the corresponding *STAT_ANALYSIS_JOB_NAME*. If using -dump_row, use -dump_row [dump_row_filename]. If using -out_stat, -out_stat [out_stat_filename]. For more information on these job arguments, please see the MET User's Guide.

Used by: StatAnalysis

STAT_ANALYSIS_JOB_NAME

Specify stat_analysis job name to run. Valid options are filter, summary, aggregate, aggregate_stat, go_index, and ramp. For more information on these job names and what they do, please see the MET User's Guide.

Used by: StatAnalysis

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 68) for more information

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_FILES_INPUT_DIR

> **Warning: DEPRECATED:** Please use *MAKE_PLOTS_INPUT_DIR* instead.

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_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 68) 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 nth 'consensus.members' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_CONSENSUS<n>_MIN_REQ

Specify the value for nth 'consensus.min_req' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_CONSENSUS<n>_NAME
Specify the value for nth 'consensus.name' in the MET configuration file for TCPairs.

Used by: TCPairs

TC_PAIRS_CONSENSUS<n>_REQUIRED
Specify the value for nth 'consensus.required' 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_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_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 68) 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 and LOOP_ORDER = processes, 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. If LOOP_ORDER = times, then TCPairs will still run for each run time. 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 68) 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_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_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_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_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 68) for more information

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 51) 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 28) 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 28) 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 28) 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 28) 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: MakePlots, StatAnalysis

VAR<n>_WAVE_NUM_LIST

Specify a comma separated list of wave numbers pairings of the Fourier decomposition.

Used by: MakePlots, StatAnalysis

VAR_LIST

> **Warning: DEPRECATED:** Please use *FCST_VAR<n>_NAME* and *OBS_VAR<n>_NAME* instead.

VERIF_CASE

> **Warning: DEPRECATED:** Please use *MAKE_PLOTS_VERIF_CASE* instead.

VERIF_GRID

> **Warning: DEPRECATED:** Please use *MAKE_PLOTS_VERIF_GRID* instead.

VERIF_TYPE

> **Warning: DEPRECATED:** Please use *MAKE_PLOTS_VERIF_TYPE* 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.

Used by: MakePlots, 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 8

METplus Statistics & Diagnostics

8.1 Statistics Database

8.1.1 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

8.1.2 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	Con-	Point-Stat Grid-	CNT SSVAR

8.1.3 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	Cate-	Grid-Stat	DMAP

8.1.4 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

8.1.5 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

8.1.6 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

8.1.7 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

8.1.8 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($(uo\text{-}uc)^2+(vo\text{-}vc)^2$)	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_BASER	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	Contin-	Point-Stat	VL1L2

8.2 Diagnostics Database

8.2.1 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

8.2.2 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

8.2.3 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

8.2.4 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	Grid-Stat	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)	INTERSECT ION_AREA	Diagnostic	MODE	MODE obj
Ratio of intersection area to the lesser of the forecast and observation object areas (unitless)	INTERSECT ION_OVER _AREA	Diagnostic	MODE	MODE obj
"Volume" of object intersection	INTERSECT ION_VOLUME	Diagnostic	MTD	MTD 3D obj
The intensity scale skill score	ISC		Wavelet-Stat	ISC
The scale at which all information following applies	ISCALE		Wavelet-Stat	ISC
Joint Probability Distribution between variable	Joint PDF to Diagnose Relationship	Diagnostic	Grid-Diag	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

8.2.5 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

8.2.6 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	Diag-	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.