Model Evaluation Tools Version 9.0.3
User's Guide

Developmental Testbed Center
Boulder, Colorado

Tara Jensen, *Barbara Brown, Randy Bullock*
Tressa Fowler, John Halley Gotway, and Kathryn Newman
with contributions from Julie Prestopnik, Eric Gilleland, Howard Soh,
Minna Win-Gildenmeister, George McCabe, James Frimel, David Fillmore, and Lindsay Blank

June 2020

Contents

Foreword: A note to MET users

This User's guide is provided as an aid to users of the Model Evaluation Tools (MET). MET is a set of verification tools developed by the Developmental Testbed Center (DTC) for use by the numerical weather prediction community to help them assess and evaluate the performance of numerical weather predictions.

It is important to note here that MET is an evolving software package. Previous releases of MET have occurred each year since 2008. This documentation describes the 9.0.1 bugfix release from April 2020. MET is also able to accept new modules contributed by the community. If you have code you would like to contribute, we will gladly consider your contribution. Please send email to: met_help@ucar.edu. We will then determine the maturity of new verification method and coordinate the inclusion of the new module in a future version.

This User's Guide was prepared by the developers of the MET, including Tressa Fowler, John Halley Gotway, Randy Bullock, Kathryn Newman, Julie Prestopnik, Lisa Goodrich, Tara Jensen, Barbara Brown, Howard Soh, Tatiana Burek, Minna Win-Gildenmeister, George McCabe, David Fillmore, Paul Prestopnik, Eric Gilleland, Nancy Rehak, Paul Oldenburg, Anne Holmes, Lacey Holland, David Ahijevych and Bonny Strong.

Bugfixes for MET v9.0

Each of these release notes is followed by the GitHub issue number which describes the bugfix.

Bugfixes in v9.0.3: https://github.com/NCAR/MET/milestone/66?closed=1

- Fix support for fractional days in NetCDF files (#1370).
- Fix runtime issues for regrid_data_plane (#1363).
- Fix Fractions Skill Score aggregation for missing data values (#1362).
- Fix Ensemble-Stat NMEP configuration options (#1351).

Bugfixes in v9.0.2: https://github.com/NCAR/MET/milestone/65?closed=1

- Fix Ensemble-Stat runtime error when requesting only RHIST, PHIST, or RELP output line types (#1342).
- Fix Grid-Stat to support MAXGAUSS smoothing method (#1335).
- Check for bad data when computing the Gerrity Score (#1335).
- Fix ascii2nc to compile without support for Python embedding (#1335).
- Correct omissions in the MET User's Guide (#1335).

Bugfixes in v9.0.1: https://github.com/NCAR/MET/milestone/64?closed=1

- Correct the definition of ensemble spread (#1294).
  - NOTE: This changes the spread statistics computed by MET!
- Fix ascii2nc python embedding with observation variable names (#1306).
- Fix python3_script.cc compilation error on a Mac (#1281).
- Fix PB2NC memory corruption bug (#1286).
- Fix point2grid segfault (#1298).

New for MET v9.0

MET version 9.0 includes some major enhancements. For Python embedding, these include the transition from Python 2 to 3, adding support in ASCII2NC and Stat-Analysis, supporting multiple input files in Ensemble-Stat, Series-Analysis, and MTD, supporting pandas, and handling the user's Python environment. Additional enhancements include the application of binned climatologies, the computation of the Ranked Probability Score (RPS) and Distance Map (DMAP) output lines types, and the addition of five new tools: Grid-Diag, Point2Grid, TC-Gen, TC-RMW, and RMW-Analysis.

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

Output Format Changes:

- Add new ensemble Ranked Probability Score (RPS) line type to the output of Ensemble-Stat and Point-Stat (for HiRA) (#681).
- Add MTD header columns for "FCST_CONV_TIME_BEG", "FCST_CONV_TIME_END", "OBS_CONV_TIME_BEG", and "OBS_CONV_TIME_END" (#1133).
- Add MTD data column for a user-specified intensity percentile value INTENSITY_*, where * is the user-specified percentile (#1134).

Configuration File Changes:

- Climatology Settings
  - Add the "climo_stdev" and "climo_cdf" dictionaries for binned climatology logic (#1224).
  - Replace the "climo_mean" dictionary options for "match_day" and "time_step" with "day_interval" and "hour_interval" (#1138).
  - Replace the "climo_cdf_bins" integer option with the "climo_cdf" dictionary (#545).
- Ensemble-Stat
  - Add the "nbrhd_prob" and "nmep_smooth" dictionaries for computing neighborhood ensemble probability forecasts (#1089).
  - Add the "nep" and "nemp" entries to the "ensemble_flag" dictionary (#1089).
  - Add the "rps" entry to the "output_flag" dictionary (#681).
  - Add the "prob_cat_thresh" option to define probability thresholds for the RPS line type (#1262).
  - Add the "sid_inc" option to specify which stations should be included in the verification (#1235).
- Grid-Stat
  - Replace the "nc_pairs_var_str" option with the "nc_pairs_var_suffix" and add the "nc_pairs_var_name" option (#1271).
  - Add the "climo_cdf" entry to the "nc_pairs_flag" dictionary (#545).
  - Add the "distance_map" dictionary to control output for the DMAP line type (#600).
  - Add the "dmap" entry to the "output_flag" dictionary (#600).
  - Add the "distance_map" entry to the "nc_pairs_flag" dictionary (#600).
- Point-Stat
  - Add the "sid_inc" option to specify which stations should be included in the verification (#1235).
  - Add the "prob_cat_thresh" entry to the "hira" dictionary (#1262).
  - Add the "rps" entry to the "output_flag" dictionary (#681).
- Series-Analysis
  - Add the "climo_stdev" dictionary to support CDP thresholds (#1138).
- MTD
  - Add the "conv_time_window" dictionary to the "fcst" and "obs" dictionaries to control the amount of temporal smoothing (#1084).
  - Add the "inten_perc_value" option to specify the desired intensity percentile to be reported (#1134).
- Point2Grid, Grid-Diag, TC-Gen, TC-RMW, RMW-Analysis
  - Add default configuration files for these new tools.

Build Process Changes:

- Transition MET source code and issue tracking from Subversion and Jira to GitHub (#805).
- Enable the G2C library archive file name to be specified at configuration time by setting "GRIB2CLIB_NAME" (default is libgrib2c.a) (#1240).
- Enable the BUFRLIB library archive file name to be specified at configuration time by setting "BUFRLIB_NAME" (default is libbufr.a) (#1185).
- Update the copyright date to 2020 and switch to the Apache 2.0 license (#1230).
- Integrate the Dockerfile into MET GitHub repository and automatically build the master_v8.1 branch, the develop branch, and all tagged releases on DockerHub (#1123).
- Document the option to install MET into "exec" rather than "bin" (#1189).
- Continued tracking and reduction of Fortify findings.

Enhancements to Existing Tools:

- Changes for all bugs fixed by met-8.1.1 and met-8.1.2.
  - https://github.com/NCAR/MET/milestone/61?closed=1
  - https://github.com/NCAR/MET/milestone/60?closed=1
- Grid Library
  - Add definitions for 51 missing pre-defined NCEP grids (#893).
  - Fix bug in the handling of some pre-defined NCEP grids (#1253).
  - Fix inconsistencies for many of the pre-defined NCEP grids (#1254).
  - Fix segfault when passing as input a thinned lat/lon grid (#1252).
  - Fix for Lambert Conformal grids crossing the international date line (#1276).
- Python Embedding
  - Switch Python embedding from Python 2 to Python 3 (#1080).
  - Enhance Python embedding to support multiple input data types (#1056).
  - Restructure the Python embedding logic to check for the "MET_PYTHON_EXE" environment variable and run the user-specified instance of Python to write a temporary pickle file (#1205).
  - Refine and simplify the Python embedding pickle logic by testing on NOAA machines, Hera and WCOSS (#1264).
  - For Python embedding, support the use of the "MET_PYTHON_INPUT_ARG" constant (#1260).
- NetCDF and GRIB Libraries
  - Fix bug in processing CF Compliant NetCDF valid time stamps (#1238).

  - Update the vx_data2d_nccf library to support all documented variants of time units (#1245).
  - Fix bug to allow for negative values of unixtime, prior to 1/1/1970 (#1239).
  - Add "GRIB1_tri" configuration file option to filter GRIB1 records based on the time range indicator value (#1263).
  - Bugfix for reporting the units for GRIB2 probabilities as "%" (#1212).

- Common Libraries
  - Print a warning message when a user specifies a config file entry as the wrong type (#1225).
  - Fix bug in the parsing of file lists and make this logic consistent across Series-Analysis, Ensemble-Stat, MTD, and TC-RMW (#1226).
  - When the climo mean and/or standard deviation fields contain bad data, exclude that matched pair from the verification (#1204).
  - Break out the Gaussian algorithm into "GAUSSIAN" and "MAXGAUSS" where "GAUSSIAN" applies a Gaussian filter using the "gaussian_dx" and "gaussian_radius" options while "MAXGAUSS" computes the maximum over the neighborhood prior to applying the Gaussian filter (#1234).
  - Report AW_MEAN regridding width at 1, not NA (#1186).
  - Add support for climatological distribution percentile thresholds, such as >CDP50 (#1138).
  - Fix MET-TC bug in the computation of initialization hour and valid hour (#1227).

- PB2NC
  - Add the derivation of PBL and ensure consistency with VSDB (#1199).
  - Remove non-printable characters that are included in the output of the "-index" command line option (#1241).

- ASCII2NC
  - Enhance ascii2nc to read point observations via Python embedding with the new "-format python" command line option (#1122).

- Point2Grid
  - Initial release of the new point to grid tool (#1078).
  - Enhance to process GOES16/17 smoke and dust data from ADP files (#1194).
  - Update quality control processing logic (#1168).
  - Derive AOD at 550nm from 440 and 675 (#1121).

- Regrid-Data-Plane
  - Remove GOES16/17 data processing since it was reimplemented in the new Point2Grid tool (#1243).
  - Add support for Gaussian regridding method to support the definition of surrogate-severe forecasts (#1136).

- PCP-Combine
  - Support multiple arguments for the "-pcpdir" command line option (#1191).
  - Fix bug in the handling of bad data for the "-subtract" command (#1255).
- Point-Stat
  - Enhance the HiRA logic to support CDP threshold types (#1251).
  - Add new ensemble Ranked Probability Score (RPS) output line type for HiRA (#681).
- Grid-Stat
  - Add an "nc_pairs_var_name" configuration file option to explicitly define the NetCDF matched pairs output variable names (#1271).
  - Add new distance map (DMAP) output line type (#600).
- Ensemble-Stat
  - Enhance to support Python embedding with "MET_PYTHON_INPUT_ARG" (#1140).
  - Add the computation of neighborhood probability forecasts (#1089).
  - Apply binned climatology logic using the "climo_cdf" config file option to the computation of ECNT statistics (#1224).
  - Fix logic for computing the lead time of a time-lagged ensemble to use the minimum lead time of the ensemble members (#1244).
  - Fix bug for initializing output variables when the first field processed contains missing data (#1242).
  - Add new ensemble Ranked Probability Score (RPS) output line type (#681).
- Point-Stat and Ensemble-Stat
  - Add the new "sid_inc" configuration option to explicitly specify which stations should be included in the verification (#1235).
- Point-Stat, Grid-Stat, and Ensemble-Stat
  - When applying climatology bins, report the mean of statistics across the bins for SL1L2, SAL1L2, CNT, PSTD, and ECNT line types (#1138).
- Stat-Analysis
  - Add support for evaluating point forecasts by reading matched pairs via Python embedding (#1143).
- MODE
  - Fix bug in the computation of the aspect ratio of objects with an area of 1 (#1215).
- MTD
  - Enhance to support Python embedding with "MET_PYTHON_INPUT_ARG" (#1140).

  - Make the amount of temporal smoothing a configurable option (#1084).
  - Add a user-specified object intensity percentile to the output (#1134).
  - Fix bug for the centroid longitude being reported in degrees west rather than degrees east (#1214).

- Series-Analysis
  - Enhance to support Python embedding with "MET_PYTHON_INPUT_ARG" (#1140).
  - Fix the memory allocation logic to dramatically reduce memory usage by up to a factor of 30 (#1267).

- Grid-Diag
  - Initial release of the new grid diagnostics tool (#1106).
  - Fix bug in the application of the masking regions (#1261).

- TC-Gen
  - Initial release of the new TC genesis tool (#1127).
  - Fix bug when checking the "min_duration", update log messages, and refine configuration file options (#1127).

- TC-RMW
  - Initial version of the Tropical Cyclone, Radius of Maximum Winds tool (#1085).

- RMW-Analysis
  - Initial version of the Radius of Maximum Winds Analysis tool (#1178).

TERMS OF USE

IMPORTANT!

The following notice shall be displayed on any scholarly works associated with, related to or derived from the Software:
"Model Evaluation Tools (MET) was developed at the National Center for Atmospheric Research (NCAR) through grants from the National Science Foundation (NSF), the National Oceanic and Atmospheric Administration (NOAA), the United States Air Force (USAF), and the United States Department of Energy (DOE). NCAR is sponsored by the United States National Science Foundation."

By using or downloading the Software, you agree to be bound by the terms and conditions of this Agreement.

The citation for this User's Guide should be:

T. Jensen, Brown, B., R. Bullock, T. Fowler, J. Halley Gotway, K. Newman, 2020:
The Model Evaluation Tools v9.0.2 (METv9.0.2) User's Guide. Developmental Testbed Center.
Available at:
https://dtcenter.org/sites/default/files/community-code/met/docs/user-guide/MET_Users_Guide_v9.0.2.pdf
481 pp.

Acknowledgments

We thank the the National Science Foundation (NSF) along with three organizations within the National Oceanic and Atmospheric Administration (NOAA): 1) Office of Atmospheric Research (OAR); 2) Next Generation Global Prediction System project (NGGPS); and 3) United State Weather Research Program (USWRP), the United States Air Force (USAF), and the United States Department of Energy (DOE) for their support of this work. Funding for the development of MET-TC is from the NOAA's Hurricane Forecast Improvement Project (HFIP) through the Developmental Testbed Center (DTC). Funding for the expansion of capability to address many methods pertinent to global and climate simulations was provided by NOAA's Next Generation Global Prediction System (NGGPS) and NSF Earth System Model 2 (EaSM2) projects. We would like to thank James Franklin at the National Hurricane Center (NHC) for his insight into the original development of the existing NHC verification software. Thanks also go to the staff at the Developmental Testbed Center for their help, advice, and many types of support. We released METv1.0 in January 2008 and would not have made a decade of cutting-edge verification support without those who participated in the original MET planning workshops and the now dis-banded verification advisory group (Mike Baldwin, Matthew Sittel, Elizabeth Ebert, Geoff DiMego, Chris Davis, and Jason Knievel).

The National Center for Atmospheric Research (NCAR) is sponsored by NSF. The DTC is sponsored by the National Oceanic and Atmospheric Administration (NOAA), the United States Air Force, and the National Science Foundation (NSF). NCAR is sponsored by the National Science Foundation (NSF).

Chapter 1

Overview of MET

1.1 Purpose and organization of the User's Guide

The goal of this User's Guide is to provide basic information for users of the Model Evaluation Tools (MET) to enable them to apply MET to their datasets and evaluation studies. MET was originally designed for application to the post-processed output of the Weather Research and Forecasting (WRF) model (see `http://www.wrf-model.org/index.php` for more information about the WRF). However, MET may also be used for the evaluation of forecasts from other models or applications if certain file format definitions (described in this document) are followed.

The MET User's Guide is organized as follows. Chapter 1 provides an overview of MET and its components. Chapter 2 contains basic information about how to get started with MET - including system requirements, required software (and how to obtain it), how to download MET, and information about compilers, libraries, and how to build the code. Chapter 3 - 6 focuses on the data needed to run MET, including formats for forecasts, observations, and output. These chapters also document the reformatting and masking tools available in MET. Chapters 7 - 11 focus on the main statistics modules contained in MET, including the Point-Stat, Grid-Stat, Ensemble-Stat, Wavelet-Stat and GSI Diagnostic Tools. These chapters include an introduction to the statistical verification methodologies utilized by the tools, followed by a section containing practical information, such as how to set up configuration files and the format of the output. Chapters 12 and 13 focus on the analysis modules, Stat-Analysis and Series-Analysis, which aggregate the output statistics from the other tools across multiple cases. Chapters 15 - 17 describe a suite of object-based tools, including MODE, MODE-Analysis, and MODE-TD. Chapters 18 - 24 describe tools focused on tropical cyclones, including MET-TC Overview, TC-Dland, TC-Pairs, TC-Stat, TC-Gen, TC-RMW and RMW-Analysis. Finally, Chapter 25 includes plotting tools included in the MET release for checking and visualizing data, as well as some additional tools and information for plotting MET results. The appendices provide further useful information, including answers to some typical questions (Appendix A: How do I... ?); and links and information about map projections, grids, and polylines (Appendix B). Appendices C and D provide more information about the verification measures and confidence intervals that are provided by MET. Sample code that can be used to perform analyses on the output of MET and create particular types

of plots of verification results is posted on the MET website (`https://dtcenter.org/community-code/model-evaluation-tools-met`). Note that the MET development group also accepts contributed analysis and plotting scripts which may be posted on the MET website for use by the community. It should be noted there are References plus a List of Tables and Figures between Chapter 25 and Appendices.

The remainder of this chapter includes information about the context for MET development, as well as information on the design principles used in developing MET. In addition, this chapter includes an overview of the MET package and its specific modules.

1.2 The Developmental Testbed Center (DTC)

MET 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, 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. MET 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.

The MET package will also be available to DTC visitors and to the WRF modeling community for testing and evaluation of new model capabilities, applications in new environments, and so on.

1.3 MET goals and design philosophy

The primary goal of MET development is to provide a state-of-the-art verification package to the NWP community. By "state-of-the-art" we mean that MET will incorporate newly developed and advanced verification methodologies, including new methods for diagnostic and spatial verification and new techniques provided by the verification and modeling communities. MET also utilizes and replicates the capabilities of existing systems for verification of NWP forecasts. For example, the MET package replicates existing National Center for Environmental Prediction (NCEP) operational verification capabilities (e.g., I/O, methods, statistics, data types). MET development will take into account the needs of the NWP community - including operational centers and the research and development community. Some of the MET capabilities include traditional verification approaches for standard surface and upper air variables (e.g., Equitable Threat Score, Mean Squared Error), confidence intervals for verification measures, and spatial forecast verification methods. In the future, MET will include additional state-of-the-art and new methodologies.

The MET package has been designed to be modular and adaptable. For example, individual modules can be applied without running the entire set of tools. New tools can easily be added to the MET package due to this modular design. In addition, the tools can readily be incorporated into a larger "system" that

may include a database as well as more sophisticated input/output and user interfaces. Currently, the MET package is a set of tools that can easily be applied by any user on their own computer platform. A suite of Python scripts for low-level automation of verification workflows and plotting has been developed to assist users with setting up their MET-based verification. It is called METplus and may be obtained at `https://github.com/NCAR/METplus`.

The MET code and documentation is maintained by the DTC in Boulder, Colorado. The MET package is freely available to the modeling, verification, and operational communities, including universities, governments, the private sector, and operational modeling and prediction centers.

1.4 MET components

The major components of the MET package are represented in Figure 1.1. The main stages represented are input, reformatting, plotting, intermediate output, statistical analyses, and output and aggregation/analysis. The MET-TC package functions independently of the other MET modules, as indicated in the Figure. Each of these stages is described further in later chapters. For example, the input and output formats are discussed in 3 as well as in the chapters associated with each of the statistics modules. MET input files are represented on the far left.

The reformatting stage of MET consists of the Gen-Vx-Mask, PB2NC, ASCII2NC, Pcp-Combine, MADIS2NC, MODIS regrid, WWMCA Regrid, and Ensemble-Stat tools. The PB2NC tool is used to create NetCDF files from input PrepBUFR files containing point observations. Likewise, the ASCII2NC tool is used to create NetCDF files from input ASCII point observations. Many types of data from the MADIS network can be formatted for use in MET by the MADIS2NC tool. MODIS and WWMCA files are regridded and formatted into NetCDF files by their respective tools. These NetCDF files are then used in the statistical analysis step. The Gen-Vx-Mask and Pcp-Combine tools are optional. The Gen-Vx-Mask tool will create a bitmapped masking area in a variety of ways. The output mask can then be used to efficiently limit verification to the interior of a user specified region. The Pcp-Combine tool can be used to add, subtract, or derive fields across multiple time steps. Often it is run to accumulate precipitation amounts into the time interval selected by the user - if a user would like to verify over a different time interval than is included in their forecast or observational dataset. The Ensemble-Stat tool will combine many forecasts into an ensemble mean or probability forecast. Additionally, if gridded or point observations are included, ensemble verification statistics are produced.

Several optional plotting utilities are provided to assist users in checking their output from the data preprocessing step. Plot-Point-Obs creates a postscript plot showing the locations of point observations. This can be quite useful for assessing whether the latitude and longitude of observation stations was specified correctly. Plot-Data-Plane produces a similar plot for gridded data. For users of the MODE object based verification methods, the Plot-MODE-Field utility will create graphics of the MODE object output. Finally, WWMCA-Plot produces a plot of the raw WWMCA data file.

The main statistical analysis components of the current version of MET are: Point-Stat, Grid-Stat, Series-Analysis, Ensemble-Stat, MODE, MODE-TD (MTD), and Wavelet-Stat. The Point-Stat tool is used for

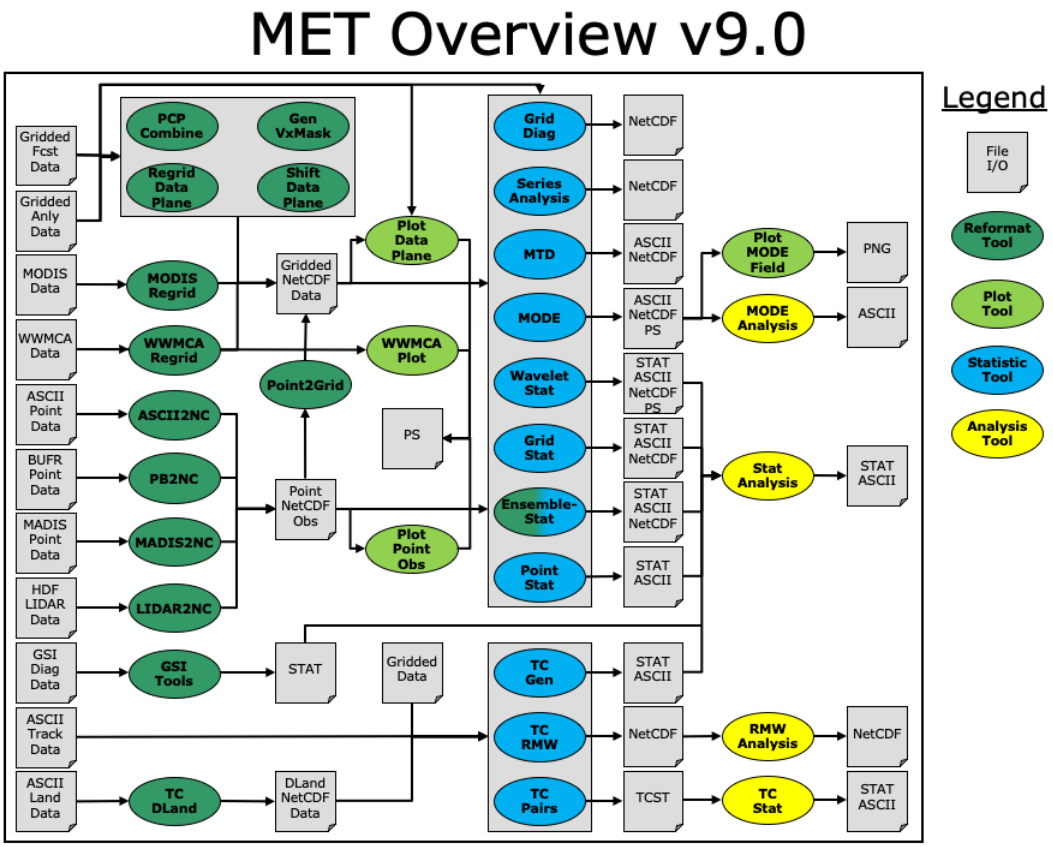

Figure 1.1: Basic representation of current MET structure and modules. Gray areas represent input and output files. Dark green areas represent reformatting and pre-processing tools. Light green areas represent plotting utilities. Blue areas represent statistical tools. Yellow areas represent aggregation and analysis tools.

grid-to-point verification, or verification of a gridded forecast field against a point-based observation (i.e., surface observing stations, ACARS, rawinsondes, and other observation types that could be described as a point observation). In addition to providing traditional forecast verification scores for both continuous and categorical variables, confidence intervals are also produced using parametric and non-parametric methods. Confidence intervals take into account the uncertainty associated with verification statistics due to sampling variability and limitations in sample size. These intervals provide more meaningful information about forecast performance. For example, confidence intervals allow credible comparisons of performance between two models when a limited number of model runs is available.

Sometimes it may be useful to verify a forecast against gridded fields (e.g., Stage IV precipitation analyses). The Grid-Stat tool produces traditional verification statistics when a gridded field is used as the observational dataset. Like the Point-Stat tool, the Grid-Stat tool also produces confidence intervals. The Grid-Stat tool also includes "neighborhood" spatial methods, such as the Fractional Skill Score (Roberts and Lean 2008). These methods are discussed in Ebert (2008). The Grid-Stat tool accumulates statistics over the entire domain.

Users wishing to accumulate statistics over a time, height, or other series separately for each grid location should use the Series-Analysis tool. Series-Analysis can read any gridded matched pair data produced by the other MET tools and accumulate them, keeping each spatial location separate. Maps of these statistics can be useful for diagnosing spatial differences in forecast quality.

The MODE (Method for Object-based Diagnostic Evaluation) tool also uses gridded fields as observational datasets. However, unlike the Grid-Stat tool, which applies traditional forecast verification techniques, MODE applies the object-based spatial verification technique described in Davis et al. (2006a,b) and Brown et al. (2007). This technique was developed in response to the "double penalty" problem in forecast verification. A forecast missed by even a small distance is effectively penalized twice by standard categorical verification scores: once for missing the event and a second time for producing a false alarm of the event elsewhere. As an alternative, MODE defines objects in both the forecast and observation fields. The objects in the forecast and observation fields are then matched and compared to one another. Applying this technique also provides diagnostic verification information that is difficult or even impossible to obtain using traditional verification measures. For example, the MODE tool can provide information about errors in location, size, and intensity.

The MODE-TD tool extends object-based analysis from two-dimensional forecasts and observations to include the time dimension. In addition to the two dimensional information provided by MODE, MODE-TD can be used to examine even more features including displacement in time, and duration and speed of moving areas of interest.

The Wavelet-Stat tool decomposes two-dimensional forecasts and observations according to the Intensity-Scale verification technique described by Casati et al. (2004). There are many types of spatial verification approaches and the Intensity-Scale technique belongs to the scale-decomposition (or scale-separation) verification approaches. The spatial scale components are obtained by applying a wavelet transformation to the forecast and observation fields. The resulting scale-decomposition measures error, bias and skill of the forecast on each spatial scale. Information is provided on the scale dependency of the error and skill, on the no-skill to skill transition scale, and on the ability of the forecast to reproduce the observed scale structure. The Wavelet-Stat tool is primarily used for precipitation fields. However, the tool can be applied to other variables, such as cloud fraction.

Though Ensemble-Stat is a preprocessing tool for creation of ensemble forecasts from a group of files, it also produces several types of ensemble statistics. Thus, it is included as a statistics tool in the flowchart.

Results from the statistical analysis stage are output in ASCII, NetCDF and Postscript formats. The Point-Stat, Grid-Stat, and Wavelet-Stat tools create STAT (statistics) files which are tabular ASCII files ending with a ".stat" suffix. In earlier versions of MET, this output format was called VSDB (Verification System DataBase). VSDB, which was developed by the NCEP, is a specialized ASCII format that can be easily read and used by graphics and analysis software. The STAT output format of the Point-Stat, Grid-Stat, and Wavelet-Stat tools is an extension of the VSDB format developed by NCEP. Additional columns of data and output line types have been added to store statistics not produced by the NCEP version.

The Stat-Analysis and MODE-Analysis tools aggregate the output statistics from the previous steps across multiple cases. The Stat-Analysis tool reads the STAT output of Point-Stat, Grid-Stat, Ensemble-Stat, and

Wavelet-Stat and can be used to filter the STAT data and produce aggregated continuous and categorical statistics. The MODE-Analysis tool reads the ASCII output of the MODE tool and can be used to produce summary information about object location, size, and intensity (as well as other object characteristics) across one or more cases.

Tropical cyclone forecasts and observations are quite different than numerical model forecasts, and thus they have their own set of tools. The MET-TC package includes several modules: TC-Dland, TC-Pairs, TC-Stat, TC-Gen, TC-RMW, and RMW-Analysis. The TC-Dland module calculates the distance to land from all locations on a specified grid. This information can be used in later modules to eliminate tropical cyclones that are over land from being included in the statistics. TC-Pairs matches up tropical cyclone forecasts and observations and writes all output to a file. In TC-Stat, these forecast / observation pairs are analyzed according to user preference to produce statistics. TC-Gen evaluates the performance of Tropical Cyclone genesis forecast using contingency table counts and statistics. TC-RMW performs a coordinate transformation for gridded model or analysis fields centered on the current storm location. RMW-Analysis filters and aggregates the output of TC-RMW across multiple cases.

The following chapters of this MET User's Guide contain usage statements for each tool, which may be viewed if you type the name of the tool. Alternatively, the user can also type the name of the tool followed by **-help** to obtain the usage statement. Each tool also has a **-version** command line option associated with it so that the user can determine what version of the tool they are using.

1.5 Future development plans

MET is an evolving verification software package. New capabilities are planned in controlled, successive version releases. Bug fixes and user-identified problems will be addressed as they are found and posted to the known issues section of the MET Users web page (`https://dtcenter.org/community-code/model-evaluation-tools-met/user-support`). Plans are also in place to incorporate many new capabilities and options in future releases of MET. Please refer to the issues listed in the MET GitHub repository (`https://github.com/NCAR/MET/issues`) to see our development priorities for upcoming releases.

1.6 Code support

MET support is provided through a MET-help e-mail address: met_help@ucar.edu. We will endeavor to respond to requests for help in a timely fashion. In addition, information about MET and tools that can be used with MET are provided on the MET Users web page (`https://dtcenter.org/community-code/model-evaluation-tools-met`).

We welcome comments and suggestions for improvements to MET, 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.

-help and **-version** command line options are available for all of the MET tools. Typing the name of the tool with no command line options also produces the usage statement.

The MET package is a "living" set of tools. Our goal is to continually enhance it and add to its capabilities. Because our time, resources, and talents are limited, we welcome contributed code for future versions of MET. These contributions may represent new verification methodologies, new analysis tools, or new plotting functions. For more information on contributing code to MET, please contact `met_help@ucar.edu`.

1.7 Fortify

Requirements from various government agencies that use MET have resulted in our code being analyzed by Fortify, a proprietary static source code analyzer owned by HP Enterprise Security Products. Fortify analyzes source code to identify for security risks, memory leaks, uninitialized variables, and other such weaknesses and bad coding practices. Fortify categorizes any issues it finds as low priority, high priority, or critical, and reports these issues back to the developers for them to address. A development cycle is thus established, with Fortify analyzing code and reporting back to the developers, who then make changes in the source code to address these issues, and hand the new code off to Fortify again for re-analysis. The goal is to drive the counts of both high priority and critical issues down to zero.

The MET developers are pleased to report that Fortify reports zero critical issues in the MET code. Users of the MET tools who work in high security environments can rest assured about the possibility of security risks when using MET, since the quality of the code has now been vetted by unbiased third-party experts. The MET developers continue using Fortify routinely to ensure that the critical counts remain at zero and to further reduce the counts for lower priority issues.

Chapter 2

Software Installation/Getting Started

2.1 Introduction

This chapter describes how to install the MET package. MET has been developed and tested on Linux operating systems. Support for additional platforms and compilers may be added in future releases. The MET package requires many external libraries to be available on the user's computer prior to installation. Required and recommended libraries, how to install MET, the MET directory structure, and sample cases are described in the following sections.

2.2 Supported architectures

The MET package was developed on Debian Linux using the GNU compilers and the Portland Group (PGI) compilers. The MET package has also been built on several other Linux distributions using the GNU, PGI, and Intel compilers. Past versions of MET have also been ported to IBM machines using the IBM compilers, but we are currently unable to support this option as the development team lacks access to an IBM machine for testing. Other machines may be added to this list in future releases as they are tested. In particular, the goal is to support those architectures supported by the WRF model itself.

The MET tools run on a single processor. Therefore, none of the utilities necessary for running WRF on multiple processors are necessary for running MET. Individual calls to the MET tools have relatively low computing and memory requirements. However users will likely be making many calls to the tools and passing those individual calls to several processors will accomplish the verification task more efficiently.

2.3 Programming languages

The MET package, including MET-TC, is written primarily in C/C++ in order to be compatible with an extensive verification code base in C/C++ already in existence. In addition, the object-based MODE and

MODE-TD verification tools relies heavily on the object-oriented aspects of C++. Knowledge of C/C++ is not necessary to use the MET package. The MET package has been designed to be highly configurable through the use of ASCII configuration files, enabling a great deal of flexibility without the need for source code modifications.

NCEP's BUFRLIB is written entirely in Fortran. The portion of MET that handles the interface to the BUFRLIB for reading PrepBUFR point observation files is also written in Fortran.

The MET package 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.

2.4 Required compilers and scripting languages

The MET package was developed and tested using the GNU g++/gfortran compilers and the Intel icc/ifort compilers. As additional compilers are successfully tested, they will be added to the list of supported platforms/compilers.

The GNU make utility is used in building all executables and is therefore required.

The MET package consists of a group of command line utilities that are compiled separately. The user may choose to run any subset of these utilities to employ the type of verification methods desired. New tools developed and added to the toolkit will be included as command line utilities.

In order to control the desired flow through MET, users are encouraged to run the tools via a script or consider using the METplus package (`https://dtcenter.org/community-code/metplus`). Some sample scripts are provided in the distribution; these examples are written in the Bourne shell. However, users are free to adapt these sample scripts to any scripting language desired.

2.5 Required libraries and optional utilities

Three external libraries are required for compiling/building MET and should be downloaded and installed before attempting to install MET. Additional external libraries required for building advanced features in MET are discussed in Section 2.6 :

1. NCEP's BUFRLIB is used by MET to decode point-based observation datasets in PrepBUFR format. BUFRLIB is distributed and supported by NCEP and is freely available for download from NCEP's website at `https://www.emc.ncep.noaa.gov/index.php?branch=BUFRLIB`. BUFRLIB requires C and Fortran-90 compilers that should be from the same family of compilers used when building MET.

2. Several tools within MET use Unidata's NetCDF libraries for writing output NetCDF files. NetCDF libraries are distributed and supported by Unidata and are freely available for download from Unidata's website at `http://www.unidata.ucar.edu/software/netcdf`. The same family of compilers used to build NetCDF should be used when building MET. MET is now compatible with the enhanced data model provided in NetCDF version 4. The support for NetCDF version 4 requires HDF5 which is freely available for download at `https://support.hdfgroup.org/HDF5/`.

3. The GNU Scientific Library (GSL) is used by MET when computing confidence intervals. GSL is distributed and supported by the GNU Software Foundation and is freely available for download from the GNU website at `http://www.gnu.org/software/gsl`.

4. The Zlib is used by MET for compression when writing postscript image files from tools (e.g. MODE, Wavelet-Stat, Plot-Data-Plane, and Plot-Point-Obs). Zlib is distributed and supported Zlib.org and is freely available for download from the Zlib website at `http://www.zlib.net`.

Two additional utilities are strongly recommended for use with MET:

1. The Unified Post-Processor is recommended for post-processing the raw WRF model output prior to verifying the model forecasts with MET. The Unified Post-Processor is freely available for download `https://dtcenter.org/community-code/unified-post-processor-upp`. MET can read data on a standard, de-staggered grid and on pressure or regular levels in the vertical. The Unified Post-Processor outputs model data in this format from both WRF cores, the NMM and the ARW. However, the Unified Post-Processor is not strictly required as long as the user can produce gridded model output on a standard de-staggered grid on pressure or regular levels in the vertical. Two-dimensional fields (e.g., precipitation amount) are also accepted for some modules.

2. The copygb utility is recommended for re-gridding model and observation datasets in GRIB version 1 format to a common verification grid. The copygb utility is distributed as part of the Unified Post-Processor and is available from other sources as well. While earlier versions of MET required that all gridded data be placed on a common grid, MET version 5.1 added support for automated re-gridding on the fly. After version 5.1, users have the option of running copygb to regrid their GRIB1 data ahead of time or leveraging the automated regridding capability within MET.

2.6 Installation of required libraries

As described in Section 2.5, some external libraries are required for building the MET:

1. NCEP's BUFRLIB is used by the MET to decode point-based observation datasets in PrepBUFR format. Once you have downloaded and unpacked the BUFRLIB tarball, refer to the README_BUFRLIB file. When compiling the library using the GNU C and Fortran compilers, users are strongly encouraged to use the -DUNDERSCORE and -fno-second-underscore options. Compiling the BUFRLIB using the GNU compilers consists of the following 3 steps:

```
* gcc -c -DUNDERSCORE *.c
* gfortran -c -DUNDERSCORE -fno-second-underscore *.f *.F
* ar crv libbufr.a *.o
```

Compiling the BUFRLIB using the PGI C and Fortran-90 compilers consists of the following 3 steps:

```
* pgcc -c -DUNDERSCORE *.c
* pgf90 -c -DUNDERSCORE -Mnosecond_underscore *.f *.F
* ar crv libbufr.a *.o
```

Compiling the BUFRLIB using the Intel icc and ifort compilers consists of the following 3 steps:

```
* icc -c -DUNDERSCORE *.c
* ifort -c -DUNDERSCORE *.f *.F
* ar crv libbufr.a *.o
```

In the directions above, the static library file that is created will be named libbufr.a. MET will check for the library file named libbufr.a, however in some cases (e.g. where the BUFRLIB is already available on a system) the library file may be named differently (e.g. libbufr_v11.3.0_4_64.a). If the library is named anything other than libbufr.a, users will need to tell MET what library to link with by passing the BUFRLIB_NAME option to MET when running configure (e.g. BUFRLIB_NAME=-lbufr_v11.3.0_4_64).

2. Unidata's NetCDF libraries are used by several tools within MET for writing output NetCDF files. The same family of compilers used to build NetCDF should be used when building MET. Users may also find some utilities built for NetCDF such as ncdump and ncview useful for viewing the contents of NetCDF files. Detailed installation instructions are available from Unidata at `http://www.unidata.ucar.edu/software/netcdf/docs/netcdf-install/`. Support for NetCDF version 4 requires HDF5. Detailed installation instructions for HDF5 are available at `https://support.hdfgroup.org/HDF5/release/obtainsrc.html`.

3. The GNU Scientific Library (GSL) is used by MET for random sampling and normal and binomial distribution computations when estimating confidence intervals. Precompiled binary packages are available for most GNU/Linux distributions and may be installed with root access. When installing GSL from a precompiled package on Debian Linux, the developer's version of GSL must be used; otherwise, use the GSL version available from the GNU website (`http://www.gnu.org/software/gsl/`). MET requires access to the GSL source headers and library archive file at build time.

4. For users wishing to compile MET with GRIB2 file support, NCEP's GRIB2 Library in C (g2clib) must be installed, along with jasperlib, libpng, and zlib. (`http://www.nco.ncep.noaa.gov/pmb/codes/GRIB2`). Please note that compiling the GRIB2C library with the **-D__64BIT__** option requires that MET also be configured with **CFLAGS="-D__64BIT__"**. Compiling MET and the GRIB2C library inconsistently may result in a segmentation fault when reading GRIB2 files. MET looks for the GRIB2C library to be named libgrib2c.a, which may be set in the GRIB2C makefile as **LIB=libgrib2c.a.** However in some cases, the library file may be named differently (e.g. libg2c_v1.6.0.a). If the library is named anything other than libgrib2c.a, users will need to tell MET

what library to link with by passing the GRIB2CLIB_NAME option to MET when running configure (e.g. GRIB2CLIB_NAME=-lg2c_v1.6.0).

5. Users wishing to compile MODIS-regrid and/or lidar2nc will need to install both the HDF4 and HDF-EOS2 libraries available from the HDF group websites (`http://www.hdfgroup.org/products/hdf4`) and (`http://www.hdfgroup.org/hdfeos.html`).

6. The MODE-Graphics utility requires Cairo and FreeType. Thus, users who wish to compile this utility must install both libraries, available from (`http://cairographics.org/releases`) and (`http://www.freetype.org/download.html`). In addition, users will need to download Ghostscript font data required at runtime (`http://sourceforge.net/projects/gs-fonts`).

2.7 Installation of optional utilities

As described in the introduction to this chapter, two additional utilities are strongly recommended for use with MET.

1. The Unified Post-Processor is recommended for post-processing the raw WRF model output prior to verifying the data with MET. The Unified Post-Processor may be used on WRF output from both the ARW and NMM cores. `https://dtcenter.org/community-code/unified-post-processor-upp` .

2. The copygb utility is recommended for re-gridding model and observation datasets in GRIB format to a common verification grid. The copygb utility is distributed as part of the Unified Post-Processor and is available from other sources as well. Please refer to the "Unified Post-processor" utility mentioned above for information on availability and installation.

2.8 MET directory structure

The top-level MET directory consists of a README file, Makefiles, configuration files, and several subdirectories. The top-level Makefile and configuration files control how the entire toolkit is built. Instructions for using these files to build MET can be found in Section 2.9.

When MET has been successfully built and installed, the installation directory contains two subdirectories. The bin/ directory contains executables for each module of MET as well as several plotting utilities. The share/met/ directory contains many subdirectories with data required at runtime and a subdirectory of sample R scripts utilities. The colortables/, map/, and ps/ subdirectories contain data used in creating PostScript plots for several MET tools. The poly/ subdirectory contains predefined lat/lon polyline regions for use in selecting regions over which to verify. The polylines defined correspond to verification regions used by NCEP as described in Appendix B. The config/ directory contains default configuration files for the MET tools. The table_files/ and tc_data/ subdirectories contain GRIB table definitions and tropical cyclone data, respectively. The Rscripts/ subdirectory contains a handful of plotting graphic utilities for

MET-TC. These are the same Rscripts that reside under the top-level MET scripts/Rscripts directory, other than it is the installed location.

The data/ directory contains several configuration and static data files used by MET. The sample_fcst/ and sample_obs/ subdirectories contain sample data used by the test scripts provided in the scripts/ directory.

The doc/ directory contains documentation for MET, including the MET User's Guide.

The out/ directory will be populated with sample output from the test cases described in the next section.

The src/ directory contains the source code for each of the tools in MET.

The scripts/ directory contains test scripts that are run by make test after MET has been successfully built, and a directory of sample configuration files used in those tests located in the scripts/config/ subdirectory. The output from the test scripts in this directory will be written to the out/ directory. Users are encouraged to copy sample configuration files to another location and modify them for their own use.

The share/met/Rscripts directory contains a handful of sample R scripts, include **plot_tcmpr.R**, which provides graphic utilities for MET-TC. For more information on the graphics capabilities, see Section 25.2.3 of this User's Guide.

2.9 Building the MET package

Building the MET package consists of three main steps: (1) install the required libraries, (2) configure the environment variables, and (3) configure and execute the build.

Install the required libraries.

- Please refer to Section 2.6 and 2.7 on how to install the required and optional libraries.
- If installing the required and optional libraries in a non-standard location, the user may need to tell MET where to find them. This can be done by setting or adding to the LD_LIBRARY PATH to included the path to the library files.

Set Environment Variables

The MET build uses environment variables to specify the locations of the needed external libraries. For each library, there is a set of three environment variables to describe the locations: $MET_<lib>, $MET_<lib>INC and $MET_<lib>LIB.

The $MET_<lib> environment variable can be used if the external library is installed such that there is a main directory which has a subdirectory called "lib" containing the library files and another subdirectory called "include" containing the include files. For example, if the NetCDF library files are installed in

/opt/netcdf/lib and the include files are in /opt/netcdf/include, you can just define the $MET_NETCDF environment variable to be "/opt/netcdf".

The $MET_<lib>INC and $MET_<lib>LIB environment variables are used if the library and include files for an external library are installed in separate locations. In this case, both environment variables must be specified and the associated $MET_<lib> variable will be ignored. For example, if the NetCDF include files are installed in /opt/include/netcdf and the library files are in /opt/lib/netcdf, then you would set $MET_NETCDFINC to "/opt/include/netcdf" and $MET_NETCDFLIB to "/opt/lib/netcdf".

The following environment variables should also be set:

- Set $MET_NETCDF to point to the main NetCDF directory, or set $MET_NETCDFINC to point to the directory with the NetCDF include files and set $MET_NETCDFLIB to point to the directory with the NetCDF library files.

- Set $MET_HDF5 to point to the main HDF5 directory.

- Set $MET_BUFR to point to the main BUFR directory, or set $MET_BUFRLIB to point to the directory with the BUFR library files. Because we don't use any BUFR library include files, you don't need to specify $MET_BUFRINC.

- Set $MET_GSL to point to the main GSL directory, or set $MET_GSLINC to point to the directory with the GSL include files and set $MET_GSLLIB to point to the directory with the GSL library files.

- If compiling support for GRIB2, set $MET_GRIB2CINC and $MET_GRIB2CLIB to point to the main GRIB2C directory which contains both the include and library files. These are used instead of $MET_GRIB2C since the main GRIB2C directory does not contain include and lib subdirectories.

- If compiling support for PYTHON, set $MET_PYTHON_CC and $MET_PYTHON_LD to specify the compiler (-I) and linker (-L) flags required for python. Set $MET_PYTHON_CC for the directory containing the "Python.h" header file. Set $MET_PYTHON_LD for the directory containing the python library file and indicate the name of that file. For example:

MET_PYTHON_CC='-I/usr/include/python3.6'

MET_PYTHON_LD='-L/usr/lib/python3.6/config-x86_64-linux-gnu -lpython3.6m'

For more information about Python support in MET, please refer to F.

- If compiling MODIS-Regrid and/or lidar2nc, set $MET_HDF to point to the main HDF4 directory, or set $MET_HDFINC to point to the directory with the HDF4 include files and set $MET_HDFLIB to point to the directory with the HDF4 library files. Also, set $MET_HDFEOS to point to the main HDF EOS directory, or set $MET_HDFEOSINC to point to the directory with the HDF EOS include files and set $MET_HDFEOSLIB to point to the directory with the HDF EOS library files.

- If compiling MODE Graphics, set $MET_CAIRO to point to the main Cairo directory, or set $MET_CAIROINC to point to the directory with the Cairo include files and set $MET_CAIROLIB to point

to the directory with the Cairo library files. Also, set $MET_FREETYPE to point to the main FreeType directory, or set $MET_FREETYPEINC to point to the directory with the FreeType include files and set $MET_FREETYPELIB to point to the directory with the FreeType library files.

- When running MODE Graphics, set $MET_FONT_DIR to the directory containing font data required at runtime. A link to the tarball containing this font data can be found on the MET website.

For ease of use, you should define these in your .cshrc or equivalent file.

Configure and execute the build

Example: To configure MET to install all of the available tools in the "bin" subdirectory of your current directory, you would use the following commands:

```
1. ./configure --prefix=‘pwd‘ --enable-grib2 --enable-python \
                --enable-modis --enable-mode_graphics --enable-lidar2nc
2. Type ’make install >& make_install.log &’
3. Type ’tail -f make_install.log’ to view the execution of the make.
4. When make is finished, type ’CNTRL-C’ to quit the tail.
```

If all tools are enabled and the build is successful, the "<prefix>/bin" directory (where <prefix> is the prefix you specified on your configure command line) will contain 36 executables:

```
- ascii2nc
- ensemble_stat
- gen_vx_mask
- grid_stat
- gis_dump_dbf
- gis_dump_shp
- gis_dump_shx
- grid_diag
- gsid2mpr
- gsidens2orank
- lidar2nc
- madis2nc
- mode
- mode_analysis
- modis_regrid
- mtd
- pb2nc
- pcp_combine
- plot_data_plane
- plot_mode_field
```

- plot_point_obs
- point2grid
- point_stat
- rmw_analysis
- regrid_data_plane
- series_analysis
- shift_data_plane
- stat_analysis
- tc_dland
- tc_gen
- tc_pairs
- tc_rmw
- tc_stat
- wavelet_stat
- wwmca_plot
- wwmca_regrid

NOTE: Several compilation warnings may occur which are expected. If any errors occur, please refer to the appendix on troubleshooting for common problems.

-help and **-version** command line options are available for all of the MET tools. Typing the name of the tool with no command line options also produces the usage statement.

The configure script has command line options to specify where to install MET and which MET utilities to install. Include any of the following options that apply to your system:

--prefix=PREFIX

By default, MET will install all the files in "/usr/local/bin". You can specify an installation prefix other than "/usr/local" using "--prefix", for instance "--prefix=$HOME" or "--prefix=`pwd`".

--enable-grib2

Enable compilation of utilities using GRIB2. Requires $MET_GRIB2C.

--enable-python

Enable compilation of python interface. Requires $MET_PYTHON_CC and $MET_PYTHON_LD.

--disable-block4

Disable use of BLOCK4 in the compilation. Use this if you have trouble using PrepBUFR files.

Run the configure script with the **--help** argument to see the full list of configuration options.

Make Targets

The autoconf utility provides some standard make targets for the users. In MET, the following standard targets have been implemented and tested:

1. **all** - compile all of the components in the package, but don't install them.
2. **install** - install the components (where is described below). Will also compile if "make all" hasn't been done yet.
3. **clean** - remove all of the temporary files created during the compilation.
4. **uninstall** - remove the installed files. For us, these are the executables and the files in $MET_BASE.

MET also has the following non-standard targets:

5. **test** - runs the scripts/test_all.sh script. You must run "make install" before using this target.

2.10 Sample test cases

Once the MET package has been built successfully, the user is encouraged to run the sample test scripts provided. They are run using make test in the top-level directory. Execute the following commands:

1. Type 'make test >& make_test.log &' to run all of the test scripts in the directory. These test scripts use test data supplied with the tarball. For instructions on running your own data, please refer to the MET User's Guide.
2. Type 'tail -f make_test.log' to view the execution of the test script.
3. When the test script is finished, type 'CNTRL-C' to quit the tail. Look in "out" to find the output files for these tests. Each tool has a separate, appropriately named subdirectory for its output files.
4. In particular, check that the PB2NC tool ran without error. If there was an error, run "make clean" then rerun your configure command adding "--disable-block4" to your configure command line and rebuild MET.

Chapter 3

MET Data I/O

Data must often be preprocessed prior to using it for verification. Several MET tools exist for this purpose. In addition to preprocessing observations, some plotting utilities for data checking are also provided and described at the end of this chapter. Both the input and output file formats are described in this chapter. Sections 3.1 and 3.2 are primarily concerned with re-formatting input files into the intermediate files required by some MET modules. These steps are represented by the first three columns in the MET flowchart depicted in Figure 1.1. Output data formats are described in later Section 3.3. Common configuration files options are described in Section 3.5. Description of software modules used to reformat the data may now be found in Chapters 4 and 5.

3.1 Input data formats

The MET package can handle gridded input data in one of four formats: GRIB version 1, GRIB version 2, NetCDF files following the Climate and Forecast (CF) conventions, and NetCDF files produced by the MET tools themselves. MET supports standard NCEP, USAF, UKMet Office and ECMWF grib tables along with custom, user-defined GRIB tables and the extended PDS including ensemble member metadata. See 3.5.1 for more information. Point observation files may be supplied in either PrepBUFR, ASCII, or MADIS format. Note that MET does not require the Unified Post-Processor to be used, but does require that the input GRIB data be on a standard, de-staggered grid on pressure or regular levels in the vertical. While the Grid-Stat, Wavelet-Stat, MODE, and MTD tools can be run on a gridded field at virtually any level, the Point-Stat tool can only be used to verify forecasts at the surface or on pressure or height levels. MET does not interpolate between native model vertical levels.

When comparing two gridded fields with the Grid-Stat, Wavelet-Stat, Ensemble-Stat, MODE, MTD, or Series-Analysis tools, the input model and observation datasets must be on the same grid. MET will regrid files according to user specified options. Alternately, outside of MET, the copygb and wgrib2 utilities are recommended for re-gridding GRIB1 and GRIB2 files, respectively. To preserve characteristics of the

observations, it is generally preferred to re-grid the model data to the observation grid, rather than vice versa.

Input point observation files in PrepBUFR format are available through NCEP. The PrepBUFR observation files contain a wide variety of point-based observation types in a single file in a standard format. However, some users may wish to use observations not included in the standard PrepBUFR files. For this reason, prior to performing the verification step in the Point-Stat tool, the PrepBUFR file is reformatted with the PB2NC tool. In this step, the user can select various ways of stratifying the observation data spatially, temporally, and by type. The remaining observations are reformatted into an intermediate NetCDF file. The ASCII2NC tool may be used to convert ASCII point observations that are not available in the PrepBUFR files into this NetCDF format for use by the Point-Stat verification tool. Users with METAR or RAOB data from MADIS can convert these observations into NetCDF format with the MADIS2NC tool, then use them with the Point-Stat or Ensemble-Stat verification tools.

Tropical cyclone forecasts and observations are typically provided in a specific ASCII format, in A Deck and B Deck files.

3.2 Intermediate data formats

MET uses NetCDF as an intermediate file format. The MET tools which write gridded output files write to a common gridded NetCDF file format. The MET tools which write point output files write to a common point observation NetCDF file format.

3.3 Output data formats

The MET package currently produces output in the following basic file formats: STAT files, ASCII files, NetCDF files, PostScript plots, and png plots from the Plot-Mode-Field utility.

The STAT format consists of tabular ASCII data that can be easily read by many analysis tools and software packages. MET produces STAT output for the Grid-Stat, Point-Stat, Ensemble-Stat, Wavelet-Stat, and TC-Gen tools. STAT is a specialized ASCII format containing one record on each line. However, a single STAT file will typically contain multiple line types. Several header columns at the beginning of each line remain the same for each line type. However, the remaining columns after the header change for each line type. STAT files can be difficult for a human to read as the quantities represented for many columns of data change from line to line.

For this reason, ASCII output is also available as an alternative for these tools. The ASCII files contain exactly the same output as the STAT files but each STAT line type is grouped into a single ASCII file with a column header row making the output more human-readable. The configuration files control which line types are output and whether or not the optional ASCII files are generated.

The MODE tool creates two ASCII output files as well (although they are not in a STAT format). It generates an ASCII file containing contingency table counts and statistics comparing the model and observation fields being compared. The MODE tool also generates a second ASCII file containing all of the attributes for the single objects and pairs of objects. Each line in this file contains the same number of columns, and those columns not applicable to a given line type contain fill data. Similarly, the MTD tool writes one ASCII output file for 2D objects attributes and four ASCII output files for 3D object attributes.

The TC-Pairs and TC-Stat utilities produce ASCII output, similar in style to the STAT files, but with TC relevant fields.

Many of the tools generate gridded NetCDF output. Generally, this output acts as input to other MET tools or plotting programs. The point observation preprocessing tools produce NetCDF output as input to the statistics tools. Full details of the contents of the NetCDF files is found in Section 3.4 below.

The MODE, Wavelet-Stat and plotting tools produce PostScript plots summarizing the spatial approach used in the verification. The PostScript plots are generated using internal libraries and do not depend on an external plotting package. The MODE plots contain several summary pages at the beginning, but the total number of pages will depend on the merging options chosen. Additional pages will be created if merging is performed using the double thresholding or fuzzy engine merging techniques for the forecast and observation fields. The number of pages in the Wavelet-Stat plots depend on the number of masking tiles used and the dimension of those tiles. The first summary page is followed by plots for the wavelet decomposition of the forecast and observation fields. The generation of these PostScript output files can be disabled using command line options.

Users can use the optional plotting utilities Plot-Data-Plane, Plot-Point-Obs, and Plot-Mode-Field to produce graphics showing forecast, observation, and MODE object files.

3.4 Data format summary

The following is a summary of the input and output formats for each of the tools currently in MET. The output listed is the maximum number of possible output files. Generally, the type of output files generated can be controlled by the configuration files and/or the command line options:

1. **PB2NC Tool**
    * **Input**: One PrepBUFR point observation file and one configuration file.
    * **Output**: One NetCDF file containing the observations that have been retained.
2. **ASCII2NC Tool**
    * **Input**: One or more ASCII point observation file(s) that has (have) been formatted as expected, and optional configuration file.
    * **Output**: One NetCDF file containing the reformatted observations.

3. **MADIS2NC Tool**

   * **Input**: One MADIS point observation file.
   * **Output**: One NetCDF file containing the reformatted observations.

4. **LIDAR2NC Tool**

   * **Input**: One CALIPSO satellite HDF file
   * **Output**: One NetCDF file containing the reformatted observations.

5. **Point2Grid Tool**

   * **Input**: One NetCDF file containing point observation from the ASCII2NC, PB2NC, MADIS2NC, or LIDAR2NC tool.
   * **Output**: One NetCDF file containing a gridded representation of the point observations.

6. **Pcp-Combine Tool**

   * **Input**: Two or more gridded model or observation files (in GRIB format for "sum" command, or any gridded file for "add", "subtract", and "derive" commands) containing data (often accumulated precipitation) to be combined.
   * **Output**: One NetCDF file containing output for the requested operation(s).

7. **Regrid-Data-Plane Tool**

   * **Input**: One gridded model or observation field and one gridded field to provide grid specification if desired.
   * **Output**: One NetCDF file containing the regridded data field(s).

8. **Shift-Data-Plane Tool**

   * **Input**: One gridded model or observation field.
   * **Output**: One NetCDF file containing the shifted data field.

9. **MODIS-Regrid Tool**

   * **Input**: One gridded model or observation field and one gridded field to provide grid specification.
   * **Output**: One NetCDF file containing the regridded data field.

10. **Gen-VX-Mask Tool**

    * **Input**: One gridded model or observation file and one file defining the masking region (varies based on masking type).
    * **Output**: One NetCDF file containing a bitmap for the resulting masking region.

11. **Point-Stat Tool**

* **Input**: One gridded model file, at least one point observation file in NetCDF format (as the output of the PB2NC, ASCII2NC, MADIS2NC, or LIDAR2NC tool), and one configuration file.
* **Output**: One STAT file containing all of the requested line types and several ASCII files for each line type requested.

12. **Grid-Stat Tool**

* **Input**: One gridded model file, one gridded observation file, and one configuration file.
* **Output**: One STAT file containing all of the requested line types, several ASCII files for each line type requested, and one NetCDF file containing the matched pair data and difference field for each verification region and variable type/level being verified.

13. **Ensemble Stat Tool**

* **Input**: An arbitrary number of gridded model files, one or more gridded and/or point observation files, and one configuration file. Point and gridded observations are both accepted.
* **Output**: One NetCDF file containing requested ensemble forecast information. If observations are provided, one STAT file containing all requested line types, several ASCII files for each line type requested, and one NetCDF file containing gridded observation ranks.

14. **Wavelet-Stat Tool**

* **Input**: One gridded model file, one gridded observation file, and one configuration file.
* **Output**: One STAT file containing the "ISC" line type, one ASCII file containing intensity-scale information and statistics, one NetCDF file containing information about the wavelet decomposition of forecast and observed fields and their differences, and one PostScript file containing plots and summaries of the intensity-scale verification.

15. **GSID2MPR Tool**

* **Input**: One or more binary GSI diagnostic files (conventional or radiance) to be reformatted.
* **Output**: One ASCII file in matched pair (MPR) format.

16. **GSID2ORANK Tool**

* **Input**: One or more binary GSI diagnostic files (conventional or radiance) to be reformatted.
* **Output**: One ASCII file in observation rank (ORANK) format.

17. **Stat-Analysis Tool**

* **Input**: One or more STAT files output from the Point-Stat, Grid-Stat, Ensemble Stat, Wavelet-Stat, or TC-Gen tools and, optionally, one configuration file containing specifications for the analysis job(s) to be run on the STAT data.
* **Output**: ASCII output of the analysis jobs is printed to the screen unless redirected to a file using the "-out" option or redirected to a STAT output file using the "-out_stat" option.

18. **Series-Analysis Tool**

    * **Input**: An arbitrary number of gridded model files and gridded observation files and one configuration file.
    * **Output**: One NetCDF file containing requested output statistics on the same grid as the input files.

19. **Grid-Diag Tool**

    * **Input**: An arbitrary number of gridded data files and one configuration file.
    * **Output:** One NetCDF file containing individual and joint histograms of the requested data.

20. **MODE Tool**

    * **Input**: One gridded model file, one gridded observation file, and one or two configuration files.
    * **Output**: One ASCII file containing contingency table counts and statistics, one ASCII file containing single and pair object attribute values, one NetCDF file containing object indices for the gridded simple and cluster object fields, and one PostScript plot containing a summary of the features-based verification performed.

21. **MODE-Analysis Tool**

    * **Input**: One or more MODE object statistics files from the MODE tool and, optionally, one configuration file containing specification for the analysis job(s) to be run on the object data.
    * **Output**: ASCII output of the analysis jobs will be printed to the screen unless redirected to a file using the "-out" option.

22. **MODE-TD Tool**

    * **Input**: Two or more gridded model files, two or more gridded observation files, and one configuration file.
    * **Output**: One ASCII file containing 2D object attributes, four ASCII files containing 3D object attributes, and one NetCDF file containing object indices for the gridded simple and cluster object fields.

23. **TC-Dland Tool**

    * **Input**: One or more files containing the longitude (Degrees East) and latitude (Degrees North) of all the coastlines and islands considered to be a significant landmass.
    * **Output**: One NetCDF format file containing a gridded field representing the distance to the nearest coastline or island, as specified in the input file.

24. **TC-Pairs Tool**

    * **Input**: At least one A-deck and one B-deck ATCF format file containing output from a tropical cyclone tracker and one configuration file. The A-deck files contain forecast tracks while the B-deck files are typically the NHC Best Track Analysis but could also be any ATCF format reference.

* **Output**: ASCII output with the suffix .tcstat.

25. **TC-Stat Tool**

    * **Input**: One or more TCSTAT output files output from the TC-Pairs tool and, optionally, one configuration file containing specifications for the analysis job(s) to be run on the TCSTAT data.
    * **Output**: ASCII output of the analysis jobs will be printed to the screen unless redirected to a file using the "-out" option.

26. **TC-Gen Tool**

    * **Input**: One or more Tropical Cyclone genesis format files, one or more verifying operational and BEST track files in ATCF format, and one configuration file.
    * **Output**: One STAT file containing all of the requested line types and several ASCII files for each line type requested.

27. **TC-RMW Tool**

    * **Input**: One or more gridded data files, one ATCF track file defining the storm location, and one configuration file.
    * **Output**: One gridded NetCDF file containing the requested model fields transformed into cylindrical coordinates.

28. **RMW-Analysis Tool**

    * **Input**: One or more NetCDF output files from the TC-RMW tool and one configuration file.
    * **Output**: One NetCDF file for results aggregated across the filtered set of input files.

29. **Plot-Point-Obs Tool**

    * **Input**: One NetCDF file containing point observation from the ASCII2NC, PB2NC, MADIS2NC, or LIDAR2NC tool.
    * **Output**: One postscript file containing a plot of the requested field.

30. **Plot-Data-Plane Tool**

    * **Input**: One gridded data file to be plotted.
    * **Output**: One postscript file containing a plot of the requested field.

31. **Plot-MODE-Field Tool**

    * **Input**: One or more MODE output files to be used for plotting and one configuration file.
    * **Output**: One PNG file with the requested MODE objects plotted. Options for objects include raw, simple or cluster and forecast or observed objects.

32. **GIS-Util Tools**

    * **Input**: ESRI shape files ending in .dbf, .shp, or .shx.
    * **Output**: ASCII description of their contents printed to the screen.

3.5 Configuration File Details

Part of the strength of MET is the leveraging of capability across tools. There are several config options that are common to many of the tools. They are described in this section.

Many of the MET tools use a configuration file to set parameters. This prevents the command line from becoming too long and cumbersome and makes the output easier to duplicate.

Settings common to multiple tools are described in the following sections while those specific to individual tools are explained in the chapters for those tools. In addition, these configuration settings are described in the share/met/config/README file and the share/met/config/README-TC file for the MET-Tropical Cyclone tools.

3.5.1 MET Configuration File Options

The information listed below may also be found in the data/config/README file.

```
////////////////////////////////////////////////////////////////////////
//
// Configuration file overview.
//
////////////////////////////////////////////////////////////////////////

The configuration files that control many of the MET tools contain formatted
ASCII text. This format has been updated for METv4.0 and continues to be used
in subsequent releases.

Settings common to multiple tools are described in the top part of this README
file and settings specific to individual tools are described beneath the common
settings. Please refer to the MET User's Guide in the "doc" directory for more
details about the settings if necessary.

A configuration file entry is an entry name, followed by an equal sign (=),
followed by an entry value, and is terminated by a semicolon (;). The
configuration file itself is one large dictionary consisting of entries, some of
which are dictionaries themselves.

The configuration file language supports the following data types:
   - Dictionary:
      - Grouping of one or more entries enclosed by curly braces {}.
   - Array:
      - List of one or more entries enclosed by square braces [].
```

```
   - Array elements are separated by commas.
- String:
   - A character string enclosed by double quotation marks "".
- Integer:
   - A numeric integer value.
- Float:
   - A numeric float value.
- Boolean:
   - A boolean value (TRUE or FALSE).
- Threshold:
   - A threshold type (<, <=, ==, !-, >=, or >) followed by a numeric value.
   - The threshold type may also be specified using two letter abbreviations
     (lt, le, eq, ne, ge, gt).
   - Multiple thresholds may be combined by specifying the logic type of AND
     (&&) or OR (||). For example, ">=5&&<=10" defines the numbers between 5
     and 10 and "==1||==2" defines numbers exactly equal to 1 or 2.
- Percentile Thresholds:
   - Thresholds may be defined as percentiles of the data being processed in
     several places:
      - In Point-Stat and Grid-Stat when setting "cat_thresh", "wind_thresh",
        and "cnt_thresh".
      - In Wavelet-Stat when setting "cat_thresh".
      - In MODE when setting "conv_thresh" and "merge_thresh".
      - In Ensemble-Stat when setting "obs_thresh".
      - When using the "censor_thresh" config option.
      - In the Stat-Analysis "-out_fcst_thresh" and "-out_obs_thresh" job
        command options.
      - In the Gen-Vx-Mask "-thresh" command line option.
   - The following percentile threshold types are supported:
      - "SFP" for a percentile of the sample forecast values.
        e.g. ">SFP50" means greater than the 50-th forecast percentile.
      - "SOP" for a percentile of the sample observation values.
        e.g. ">SOP75" means greater than the 75-th observation percentile.
      - "SCP" for a percentile of the sample climatology values.
        e.g. ">SCP90" means greater than the 90-th climatology percentile.
      - "USP" for a user-specified percentile threshold.
        e.g. "<USP90(2.5)" means less than the 90-th percentile values which
        the user has already determined to be 2.5 outside of MET.
      - "==FBIAS1" to automatically de-bias the data. This option must be
        used in conjunction with a simple threshold in the other field.
        For example, when "obs.cat_thresh = >5.0" and
        "fcst.cat_thresh = ==FBIAS1;", MET applies the >5.0 threshold to the
        observations and then chooses a forecast threshold which results in a
```

```
          frequency bias of 1.
        - "CDP" for climatological distribution percentile thresholds.
          These thresholds require that the climatological mean and standard
          deviation be defined using the climo_mean and climo_stdev config file
          options, respectively. The categorical (cat_thresh), conditional
          (cnt_thresh), or wind speed (wind_thresh) thresholds are defined
          relative to the climatological distribution at each point. Therefore,
          the actual numeric threshold applied can change for each point.
          e.g. ">CDP50" means greater than the 50-th percentile of the
          climatological distribution for each point.
    - When percentile thresholds of type SFP, SOP, SCP, or CDP are requested
      for continuous filtering thresholds (cnt_thresh), wind speed thresholds
      (wind_thresh), or observation filtering thresholds (obs_thresh in
      ensemble_stat), the following special logic is applied. Percentile
      thresholds of type equality are automatically converted to percentile
      bins which span the values from 0 to 100.
      For example, "==CDP25" is automatically expanded to 4 percentile bins:
         >=CDP0&&<CDP25,>=CDP25&&<CDP50,>=CDP50&&<CDP75,>=CDP75&&<=CDP100
    - When sample percentile thresholds of type SFP, SOP, SCP, or FBIAS1 are
      requested, MET recomputes the actual percentile that the threshold
      represents. If the requested percentile and actual percentile differ by
      more than 5%, a warning message is printed. This may occur when the
      sample size is small or the data values are not truly continuous.
    - When percentile thresholds of type SFP, SOP, SCP, or USP are used, the
      actual threshold value is appended to the FCST_THRESH and OBS_THRESH
      output columns. For example, if the 90-th percentile of the current set
      of forecast values is 3.5, then the requested threshold "<=SFP90" is
      written to the output as "<=SFP90(3.5)".
    - When parsing FCST_THRESH and OBS_THRESH columns, the Stat-Analysis tool
      ignores the actual percentile values listed in parentheses.
  - Piecewise-Linear Function (currently used only by MODE):
    - A list of (x, y) points enclosed in parenthesis ().
    - The (x, y) points are *NOT* separated by commas.
  - User-defined function of a single variable:
    - Left side is a function name followed by variable name in parenthesis.
    - Right side is an equation which includes basic math functions (+,-,*,/),
      built-in functions (listed below), or other user-defined functions.
    - Built-in functions include:
      sin, cos, tan, sind, cosd, tand, asin, acos, atan, asind, acosd, atand,
      atan2, atan2d, arg, argd, log, exp, log10, exp10, sqrt, abs, min, max,
      mod, floor, ceil, step, nint, sign

The context of a configuration entry matters. If an entry cannot be found in
```

the expected dictionary, the MET tools recursively search for that entry in the parent dictionaries, all the way up to the top-level configuration file dictionary. If you'd like to apply the same setting across all cases, you can simply specify it once at the top-level. Alternatively, you can specify a setting at the appropriate dictionary level to have finer control over the behavior.

In order to make the configuration files more readable, several descriptive integer types have been defined in the ConfigConstants file. These integer names may be used on the right-hand side for many configuration file entries.

Each of the configurable MET tools expects a certain set of configuration entries. Examples of the MET configuration files can be found in data/config and scripts/config.

When you pass a configuration file to a MET tool, the tool actually parses up to four different configuration files in the following order:

(1) Reads share/met/config/ConfigConstants to define constants.
(2) If the tool produces PostScript output, it reads share/met/config/ConfigMapData to define the map data to be plotted.
(3) Reads the default configuration file for the tool from share/met/config.
(4) Reads the user-specified configuration file from the command line.

Many of the entries from step (3) are overwritten by the user-specified entries from step (4). Therefore, the configuration file you pass in on the command line really only needs to contain entries that differ from the defaults.

Any of the configuration entries may be overwritten by the user-specified configuration file. For example, the map data to be plotted may be included in the user-specified configuration file and override the default settings defined in the share/met/config/ConfigMapData file.

The configuration file language supports the use of environment variables. They are specified as ${ENV_VAR}, where ENV_VAR is the name of the environment variable. When scripting up many calls to the MET tools, you may find it convenient to use them. For example, when applying the same configuration to the output from multiple models, consider defining the model name as an environment variable which the controlling script sets prior to verifying the output of each model. Setting MODEL to that environment variable enables you to use one configuration file rather than maintianing many very similar ones.

An error in the syntax of a configuration file will result in an error from the MET tool stating the location of the parsing error.

The MET_BASE variable is defined in the code at compilation time as the path to the MET shared data. These are things like the default configuration files, common polygons and color scales. MET_BASE may be used in the MET configuration files when specifying paths and the appropriate path will be substituted in. If MET_BASE is defined as an environment variable, its value will be used instead of the one defined at compilation time.

The MET_OBS_ERROR_TABLE environment variable can be set to specify the location of an ASCII file defining observation error information. The default table can be found in the installed share/met/table_files/obs_error_table.txt. This observation error logic is applied in Ensemble-Stat to perturb ensemble member values and/or define observation bias corrections.

When processing point and gridded observations, Ensemble-Stat searches the table to find the entry defining the observation error information. The table consists of 15 columns and includes a header row defining each column. The special string "ALL" is interpreted as a wildcard in these files. The first 6 columns (OBS_VAR, MESSAGE_TYPE, PB_REPORT_TYPE, IN_REPORT_TYPE, INSTRUMENT_TYPE, and STATION_ID) may be set to a comma-separated list of strings to be matched. In addition, the strings in the OBS_VAR column are interpreted as regular expressions when searching for a match. For example, setting the OBS_VAR column to 'APCP_[0-9]+' would match observations for both APCP_03 and APCP_24. The HGT_RANGE, VAL_RANGE, and PRS_RANGE columns should either be set to "ALL" or "BEG,END" where BEG and END specify the range of values to be used. The INST_BIAS_SCALE and INST_BIAS_OFFSET columns define instrument bias adjustments which are applied to the observation values. The DIST_TYPE and DIST_PARM columns define the distribution from which random perturbations should be drawn and applied to the ensemble member values. See the obs_error description below for details on the supported error distributions. The last two columns, MIN and MAX, define the bounds for the valid range of the bias-corrected observation values and randomly perturbed ensemble member values. Values less than MIN are reset to the mimumum value and values greater than MAX are reset to the maximum value. A value of NA indicates that the variable is unbounded.

The MET_GRIB_TABLES environment variable can be set to specify the location of custom GRIB tables. It can either be set to a specific file name or to a directory containing custom GRIB tables files. These file names must begin with a "grib1" or "grib2" prefix and end with a ".txt" suffix. Their format must match the format used by the default MET GRIB table files, described below. The custom GRIB tables are read prior to the default tables and their settings take precedence.

```
At runtime, the MET tools read default GRIB tables from the installed
share/met/table_files directory, and their file formats are described below:

GRIB1 table files begin with "grib1" prefix and end with a ".txt" suffix.
The first line of the file must contain "GRIB1".
The following lines consist of 4 integers followed by 3 strings:
Column 1: GRIB code (e.g. 11 for temperature)
Column 2: parameter table version number
Column 3: center id (e.g. 07 for US Weather Service- National Met. Center)
Column 4: subcenter id
Column 5: variable name
Column 6: variable description
Column 7: units

References:
   http://www.nco.ncep.noaa.gov/pmb/docs/on388
   http://www.wmo.int/pages/prog/www/WMOCodes/Guides/GRIB/GRIB1-Contents.html

GRIB2 table files begin with "grib2" prefix and end with a ".txt" suffix.
The first line of the file must contain "GRIB2".
The following lines consist of 8 integers followed by 3 strings.
column 1:  Section 0 Discipline
column 2:  Section 1 Master Tables Version Number
column 3:  Section 1 Master Tables Version Number, low range of tables
column 4:  Section 1 Master Table Version Number, high range of tables
column 5:  Section 1 originating center
column 6:  Local Tables Version Number
column 7:  Section 4 Template 4.0 Parameter category
column 8:  Section 4 Template 4.0 Parameter number
column 9:  variable name
column 10: variable description
column 11: units

References:
   http://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc

////////////////////////////////////////////////////////////////////////////////
//
// Configuration settings used by the MET tools.
//
////////////////////////////////////////////////////////////////////////////////

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

```
//
// Settings common to multiple tools
//
////////////////////////////////////////////////////////////////////////

//
// The "exit_on_warning" entry in ConfigConstants may be set to true or false.
// If set to true and a MET tool encounters a warning, it will immediately exit
// with bad status after writing the warning message.
//
exit_on_warning = FALSE;

//
// The "nc_compression" entry in ConfigConstants defines the compression level
// for the NetCDF variables. Setting this option in the config file of one of
// the tools overrides the default value set in ConfigConstants. The
// environment variable MET_NC_COMPRESS overrides the compression level
// from configuration file. The command line argument "-compress n" for some
// tools overrides it.
// The range is 0 to 9.
//   - 0 is to disable the compression.
//   - 1 to 9: Lower number is faster, higher number for smaller files.
// WARNING: Selecting a high compression level may slow down the reading and
//          writing of NetCDF files within MET significantly.
//
nc_compression = 0;

//
// The "output_precision" entry in ConfigConstants defines the precision
// (number of significant decimal places) to be written to the ASCII output
// files. Setting this option in the config file of one of the tools will
// override the default value set in ConfigConstants.
//
output_precision = 5;

// The "tmp_dir" entry in ConfigConstants defines the directory for the
// temporary files. The directory must exist and be writable. The environment
// variable MET_TMP_DIR overrides the default value at the configuration file.
// Some tools override the temporary directory by the command line argument
// "-tmp_dir <diretory_name>".
tmp_dir = "/tmp";

//
```

```
// The "message_type_group_map" entry is an array of dictionaries, each
// containing a "key" string and "val" string. This defines a mapping of
// message type group names to a comma-separated list of values. This map is
// defined in the config files for PB2NC, Point-Stat, or Ensemble-Stat. Modify
// this map to define sets of message types that should be processed together as
// a group. The "SURFACE" entry must be present to define message types for
// which surface verification logic should be applied.
//

mesage_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";                     }
];

//
// The "message_type_map" entry is an array of dictionaries, each containing
// a "key" string and "val" string. This defines a mapping of input strings
// to output message types. This mapping is applied in ASCII2NC when
// converting input little_r report types to output message types. This mapping
// is also supported in PBN2NC as a way of renaming input PREPBUFR 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"; }
];

//
// The "model" entry specifies a name for the model being verified. This name
// is written to the MODEL column of the ASCII output generated. If you're
// verifying multiple models, you should choose descriptive model names (no
// whitespace) to distinguish between their output.
// e.g. model = "GFS";
//
```

```
model = "WRF";

//
// The "desc" entry specifies a user-specified description for each verification
// task. This string is written to the DESC column of the ASCII output
// generated. It may be set separately in each "obs.field" verification task
// entry or simply once at the top level of the configuration file. If you're
// verifying the same field multiple times with different quality control
// flags, you should choose description strings (no whitespace) to distinguish
// between their output.
// e.g. desc = "QC_9";
//
desc = "NA";

//
// The "obtype" entry specifies a name to describe the type of verifying gridded
// observation used. This name is written to the OBTYPE column in the ASCII
// output generated. If you're using multiple types of verifying observations,
// you should choose a descriptive name (no whitespace) to distinguish between
// their output. When verifying against point observations the point
// observation message type value is written to the OBTYPE column. Otherwise,
// the configuration file obtype value is written.
//
obtype = "ANALYS";

//
// The "regrid" entry is a dictionary containing information about how to handle
// input gridded data files. The "regrid" entry specifies regridding logic
// using the following entries:
//
//   - The "to_grid" entry may be set to NONE, FCST, OBS, a named grid, the path
//     to a gridded data file defining the grid, or an explicit grid specification
//     string.
//      - to_grid = NONE;   To disable regridding.
//      - to_grid = FCST;   To regrid observations to the forecast grid.
//      - to_grid = OBS;    To regrid forecasts to the observation grid.
//      - to_grid = "G218"; To regrid both to a named grid.
//      - to_grid = "path"; To regrid both to a grid defined by a file.
//      - to_grid = "spec"; To define a grid specified as follows:
//         - lambert Nx Ny lat_ll lon_ll lon_orient D_km R_km standard_parallel_1
//           [standard_parallel_2] N|S
//         - stereo Nx Ny lat_ll lon_ll lon_orient D_km R_km lat_scale N|S
//         - latlon Nx Ny lat_ll lon_ll delta_lat delta_lon
```

```
//         - mercator Nx Ny lat_ll lon_ll lat_ur lon_ur
//         - gaussian lon_zero Nx Ny
//
//    - The "vld_thresh" entry specifies a proportion between 0 and 1 to define
//      the required ratio of valid data points. When regridding, compute
//      a ratio of the number of valid data points to the total number of
//      points in the neighborhood. If that ratio is less than this threshold,
//      write bad data for the current point.
//
//   - The "method" entry defines the regridding method to be used.
//      - Valid regridding methods:
//         - MIN         for the minimum value
//         - MAX         for the maximum value
//         - MEDIAN      for the median value
//         - UW_MEAN     for the unweighted average value
//         - DW_MEAN     for the distance-weighted average value (weight =
//                       distance^-2)
//         - AW_MEAN     for an area-weighted mean when regridding from
//                       high to low resolution grids (width = 1)
//         - LS_FIT      for a least-squares fit
//         - BILIN       for bilinear interpolation (width = 2)
//         - NEAREST     for the nearest grid point (width = 1)
//         - BUDGET      for the mass-conserving budget interpolation
//         - FORCE       to compare gridded data directly with no interpolation
//                       as long as the grid x and y dimensions match.
//         - UPPER_LEFT  for the upper left grid point (width = 1)
//         - UPPER_RIGHT for the upper right grid point (width = 1)
//         - LOWER_RIGHT for the lower right grid point (width = 1)
//         - LOWER_LEFT  for the lower left grid point (width = 1)
//         - MAXGAUSS    to compute the maximum value in the neighborhood
//                       and apply a Gaussian smoother to the result
//
//         The BEST and GEOG_MATCH interpolation options are not valid for
//         regridding.
//
//   - The "width" entry specifies a regridding width, when applicable.
//      - width = 4;     To regrid using a 4x4 box or circle with diameter 4.
//
//   - The "shape" entry defines the shape of the neighborhood.
//     Valid values are "SQUARE" or "CIRCLE"
//
//   - The "gaussian_dx" entry specifies a delta distance for Gaussian
//     smoothing. The default is 81.271. Ignored if not Gaussian method.
```

```
//
//   - The "gaussian_radius" entry defines the radius of influence for Gaussian
//     smoothing. The default is 120. Ignored if not Gaussian method.
//
//   - The "gaussian_dx" and "gaussian_radius" settings must be in the same units,
//     such as kilometers or degress.  Their ratio (sigma = gaussian_radius /
//     gaussian_dx) determines the Guassian weighting function.
//
regrid = {
   to_grid         = NONE;
   method          = NEAREST;
   width           = 1;
   vld_thresh      = 0.5;
   shape           = SQUARE;
   gaussian_dx     = 81.271;
   gaussian_radius = 120;
}

//
// The "fcst" entry is a dictionary containing information about the field(s)
// to be verified. This dictionary may include the following entries:
//
//   - The "field" entry is an array of dictionaries, each specifying a
//     verification task. Each of these dictionaries may include:
//
//      - The "name" entry specifies a name for the field.
//
//      - The "level" entry specifies level information for the field.
//
//      - Setting "name" and "level" is file-format specific. See below.
//
//      - The "prob" entry in the forecast dictionary defines probability
//        information. It may either be set as a boolean (i.e. TRUE or FALSE)
//        or as a dictionary defining probabilistic field information.
//
//        When set as a boolean to TRUE, it indicates that the "fcst.field" data
//        should be treated as probabilities. For example, when verifying the
//        probabilistic NetCDF output of Ensemble-Stat, one could configure the
//        Grid-Stat or Point-Stat tools as follows:
//
//           fcst = {
//              field = [ { name  = "APCP_24_A24_ENS_FREQ_gt0.0";
//                          level = "(*,*)";
```

```
//                          prob  = TRUE; } ];
//          }
//
//        Setting "prob = TRUE" indicates that the "APCP_24_A24_ENS_FREQ_gt0.0"
//        data should be processed as probabilities.
//
//        When set as a dictionary, it defines the probabilistic field to be
//        used. For example, when verifying GRIB files containing probabilistic
//        data,  one could configure the Grid-Stat or Point-Stat tools as
//        follows:
//
//           fcst = {
//              field = [ { name = "PROB"; level = "A24";
//                          prob = { name = "APCP"; thresh_lo = 2.54; } },
//                        { name = "PROB"; level = "P850";
//                          prob = { name = "TMP"; thresh_hi = 273; } } ];
//           }
//
//        The example above selects two probabilistic fields. In both, "name"
//        is set to "PROB", the GRIB abbreviation for probabilities. The "level"
//        entry defines the level information (i.e. "A24" for a 24-hour
//        accumulation and "P850" for 850mb). The "prob" dictionary defines the
//        event for which the probability is defined. The "thresh_lo"
//        (i.e. APCP > 2.54) and/or "thresh_hi" (i.e. TMP < 273) entries are
//        used to define the event threshold(s).
//
//        Probability fields should contain values in the range
//        [0, 1] or [0, 100]. However, when MET encounters a probability field
//        with a range [0, 100], it will automatically rescale it to be [0, 1]
//        before applying the probabilistic verification methods.
//
//      - Set "prob_as_scalar = TRUE" to override the processing of probability
//        data. When the "prob" entry is set as a dictionary to define the
//        field of interest, setting "prob_as_scalar = TRUE" indicates that this
//        data should be processed as regular scalars rather than probabilities.
//        For example, this option can be used to compute traditional 2x2
//        contingency tables and neighborhood verification statistics for
//        probability data. It can also be used to compare two probability
//        fields directly. When this flag is set, probability values are
//        automatically rescaled from the range [0, 100] to [0, 1].
//
//      - The "convert" entry is a user-defined function of a single variable
//        for processing input data values. Any input values that are not bad
```

```
//        data are replaced by the value of this function. The convert function
//        is applied prior to regridding or thresholding. This function may
//        include any of the built-in math functions (e.g. sqrt, log10)
//        described above.
//        Several standard unit conversion functions are already defined in
//        data/config/ConfigConstants.
//        Examples of user-defined conversion functions include:
//          convert(x) = 2*x;
//          convert(x) = x^2;
//          convert(a) = log10(a);
//          convert(a) = a^10;
//          convert(t) = max(1, sqrt(abs(t)));
//          convert(x) = K_to_C(x); where K_to_C(x) is defined in
//                                  ConfigConstants
//
//      - The "censor_thresh" entry is an array of thresholds to be applied
//        to the input data. The "censor_val" entry is an array of numbers
//        and must be the same length as "censor_thresh". These arguments must
//        appear together in the correct format (threshold and number). For each
//        censor threshold, any input values meeting the threshold criteria will
//        be reset to the corresponding censor value. An empty list indicates
//        that no censoring should be performed. The censoring logic is applied
//        prior to any regridding but after the convert function. All statistics
//        are computed on the censored data. These entries may be used to apply
//        quality control logic by resetting data outside of an expected range
//        to the bad data value of -9999. These entries are not indicated in the
//        metadata of any output files, but the user can set the "desc" entry
//        accordingly.
//
//      Examples of user-defined conversion functions include:
//          censor_thresh = [ >12000 ];
//          censor_val    = [ 12000 ];
//
//      - The "cat_thresh" entry is an array of thresholds to be used when
//        computing categorical statistics.
//
//      - The "cnt_thresh" entry is an array of thresholds for filtering
//        data prior to computing continuous statistics and partial sums.
//
//      - The "cnt_logic" entry may be set to UNION, INTERSECTION, or SYMDIFF
//        and controls the logic for how the forecast and observed cnt_thresh
//        settings are combined when filtering matched pairs of forecast and
//        observed values.
```

```
//
//   - The "file_type" entry specifies the input gridded data file type rather
//     than letting the code determine it. MET determines the file type by
//     checking for known suffixes and examining the file contents. Use this
//     option to override the code's choice. The valid file_type values are
//     listed the "data/config/ConfigConstants" file and are described below.
//     This entry should be defined within the "fcst" and/or "obs" dictionaries.
//     e.g.
//     fcst = {
//        file_type = GRIB1;         // GRIB version 1
//        file_type = GRIB2;         // GRIB version 2
//        file_type = NETCDF_MET;    // NetCDF created by another MET tool
//        file_type = NETCDF_PINT;   // NetCDF created by running the p_interp
//                                   // or wrf_interp utility on WRF output.
//                                   // May be used to read unstaggered raw WRF
//                                   // NetCDF output at the surface or a
//                                   // single model level.
//        file_type = NETCDF_NCCF;   // NetCDF following the Climate Forecast
//                                   // (CF) convention.
//        file_type = PYTHON_NUMPY;  // Run a Python script to load data into
//                                   // a NumPy array.
//        file_type = PYTHON_XARRAY; // Run a Python script to load data into
//                                   // an xarray object.
//     }
//
//   - The "wind_thresh" entry is an array of thresholds used to filter wind
//     speed values when computing VL1L2 vector partial sums. Only those U/V
//     pairs that meet this wind speed criteria will be included in the sums.
//     Setting this threshold to NA will result in all U/V pairs being used.
//
//   - The "wind_logic" entry may be set to UNION, INTERSECTION, or SYMDIFF
//     and controls the logic for how the forecast and observed wind_thresh
//     settings are combined when filtering matched pairs of forecast and
//     observed wind speeds.
//
//   - The "eclv_points" entry specifies the economic cost/loss ratio points
//     to be evaluated. For each cost/loss ratio specified, the relative value
//     will be computed and written to the ECLV output line. This entry may
//     either be specified as an array of numbers between 0 and 1 or as a single
//     number. For an array, each array entry will be evaluated. For a single
//     number, all evenly spaced points between 0 and 1 will be evaluated, where
//     eclv_points defines the spacing. Cost/loss values are omitted for
//     ratios of 0.0 and 1.0 since they are undefined.
```

```
//
//   - The "init_time" entry specifies the initialization time in
//     YYYYMMDD[_HH[MMSS]]
//     format. This entry can be included in the "fcst" entry as shown below or
//     included in the "field" entry if the user would like to use different
//     initialization times for different fields.
//
//   - The "valid_time" entry specifies the valid time in YYYYMMDD[_HH[MMSS]]
//     format. This entry can be included in the "fcst" entry as shown below or
//     included in the "field" entry if the user would like to use different
//     valid times for different fields.
//
//   - The "lead_time" entry specifies the lead time in HH[MMSS]
//     format. This entry can be included in the "fcst" entry as shown below or
//     included in the "field" entry if the user would like to use different
//     lead times for different fields.
//
// It is only necessary to use the "init_time", "valid_time", and/or "lead_time"
// settings when verifying a file containing data for multiple output times.
// For example, to verify a GRIB file containing data for many lead times, you
// could use "lead_time" to specify the record to be verified.
//
// File-format specific settings for the "field" entry:
//
//    - GRIB1 and GRIB2:
//       - For custom GRIB tables, see note about MET_GRIB_TABLES.
//       - The "name" entry specifies a GRIB code number or abbreviation.
//         - GRIB1 Product Definition Section:
//           http://www.nco.ncep.noaa.gov/pmb/docs/on388/table2.html
//         - GRIB2 Product Definition Section:
//           http://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc
//       - The "level" entry specifies a level type and value:
//          - ANNN for accumulation interval NNN
//          - ZNNN for vertical level NNN
//          - ZNNN-NNN for a range of vertical levels
//          - PNNN for pressure level NNN in hPa
//          - PNNN-NNN for a range of pressure levels in hPa
//          - LNNN for a generic level type
//          - RNNN for a specific GRIB record number
//       - The "GRIB_lvl_typ" entry is an integer specifying the level type.
//       - The "GRIB_lvl_val1" and "GRIB_lvl_val2" entries are floats specifying
//         the first and second level values.
//       - The "GRIB_ens" entry is a string specifying NCEP's usage of the
```

```
//         extended PDS for ensembles. Set to "hi_res_ctl", "low_res_ctl",
//         "+n", or "-n", for the n-th ensemble member.
//       - The "GRIB1_ptv" entry is an integer specifying the GRIB1 parameter
//         table version number.
//       - The "GRIB1_code" entry is an integer specifying the GRIB1 code (wgrib
//         kpds5 value).
//       - The "GRIB1_center" is an integer specifying the originating center.
//       - The "GRIB1_subcenter" is an integer specifying the originating
//         subcenter.
//       - The "GRIB1_tri" is an integer specifying the time range indicator.
//       - The "GRIB2_mtab" is an integer specifying the master table number.
//       - The "GRIB2_ltab" is an integer specifying the local table number.
//       - The "GRIB2_disc" is an integer specifying the GRIB2 discipline code.
//       - The "GRIB2_parm_cat" is an integer specifying the parameter category
//         code.
//       - The "GRIB2_parm" is an integer specifying the parameter code.
//       - The "GRIB2_pdt" is an integer specifying the product definition
//         template (Table 4.0).
//       - The "GRIB2_process" is an integer specifying the generating process
//         (Table 4.3).
//       - The "GRIB2_cntr" is an integer specifying the originating center.
//       - The "GRIB2_ens_type" is an integer specifying the ensemble type
//         (Table 4.6).
//       - The "GRIB2_der_type" is an integer specifying the derived product
//         type (Table 4.7).
//       - The "GRIB2_stat_type" is an integer specifying the statistical
//         processing type (Table 4.10).
//       - The "GRIB2_ipdtmpl_index" and "GRIB2_ipdtmpl_val" entries are arrays
//         of integers which specify the product description template values to
//         be used. The indices are 0-based. For example, use the following to
//         request a GRIB2 record whose 9-th and 27-th product description
//         template values are 1 and 2, respectively:
//            GRIB2_ipdtmpl_index=[8, 26]; GRIB2_ipdtmpl_val=[1, 2];
//
//    - NetCDF (from MET tools, CF-compliant, p_interp, and wrf_interp):
//       - The "name" entry specifies the NetCDF variable name.
//       - The "level" entry specifies the dimensions to be used:
//          - (i,...,j,*,*) for a single field, where i,...,j specifies fixed
//            dimension values and *,* specifies the two dimensions for the
//            gridded field.
//       e.g.
//           field = [
//             {
```

```
//               name       = "QVAPOR";
//               level      = "(0,5,*,*)";
//             },
//             {
//               name       = "TMP_P850_ENS_MEAN";
//               level      = [ "(*,*)" ];
//             }
//
//           ];
//
//    - Python (using PYTHON_NUMPY or PYTHON_XARRAY):
//       - The Python interface for MET is described in Appendix F of the MET
//         User's Guide.
//       - Two methods for specifying the Python command and input file name
//         are supported. For tools which read a single gridded forecast and/or
//         observation file, both options work. However, only the second option
//         is supported for tools which read multiple gridded data files, such
//         as Ensemble-Stat, Series-Analysis, and MTD.
//
//       Option 1:
//          - On the command line, replace the path to the input gridded data
//            file with the constant string PYTHON_NUMPY or PYTHON_XARRAY.
//          - Specify the configuration "name" entry as the Python command to be
//            executed to read the data.
//          - The "level" entry is not required for Python.
//          e.g.
//             field = [
//               { name = "read_ascii_numpy.py data/python/fcst.txt FCST"; }
//             ];
//
//       Option 2:
//          - On the command line, leave the path to the input gridded data
//            as is.
//          - Set the configuration "file_type" entry to the constant
//            PYTHON_NUMPY or PYTHON_XARRAY.
//          - Specify the configuration "name" entry as the Python command to be
//            executed to read the data, but replace the input gridded data file
//            with the constant MET_PYTHON_INPUT_ARG.
//          - The "level" entry is not required for Python.
//          e.g.
//             file_type = PYTHON_NUMPY;
//             field     = [
//               { name = "read_ascii_numpy.py MET_PYTHON_INPUT_ARG FCST"; }
```

```
//             ];
//
fcst = {
   censor_thresh = [];
   censor_val    = [];
   cnt_thresh    = [ NA ];
   cnt_logic     = UNION;
   wind_thresh   = [ NA ];
   wind_logic    = UNION;
   eclv_points   = 0.05;
   message_type  = [ "ADPSFC" ];
   init_time     = "20120619_12";
   valid_time    = "20120620_00";
   lead_time     = "12";

   field = [
      {
        name       = "APCP";
        level      = [ "A03" ];
        cat_thresh = [ >0.0, >=5.0 ];
      }
   ];
}

//
// The "obs" entry specifies the same type of information as "fcst", but for
// the observation data. It will often be set to the same things as "fcst",
// as shown in the example below. However, when comparing forecast and
// observation files of different format types, this entry will need to be set
// in a non-trivial way. The length of the "obs.field" array must match the
// length of the "fcst.field" array.
//     e.g.
//         obs = fcst;
//
//     or
//
//         fcst = {
//           censor_thresh = [];
//           censor_val    = [];
//           cnt_thresh    = [ NA ];
//           cnt_logic     = UNION;
//           wind_thresh   = [ NA ];
//           wind_logic    = UNION;
```

```
//
//          field = [
//             {
//                name       = "PWAT";
//                level      = [ "L0" ];
//                cat_thresh = [ >2.5 ];
//             }
//           ];
//        }
//
//
//        obs = {
//          censor_thresh = [];
//          censor_val    = [];
//          cnt_thresh    = [ NA ];
//          cnt_logic     = UNION;
//          wind_thresh   = [ NA ];
//          wind_logic    = UNION;
//
//          field = [
//             {
//                name       = "IWV";
//                level      = [ "L0" ];
//                cat_thresh = [ >25.0 ];
//             }
//           ];
//        }
//
//
//   - The "message_type" entry is an array of point observation message types
//     to be used. This only applies to the tools that verify against point
//     observations. This may be specified once at the top-level "obs"
//     dictionary or separately for each "field" array element. In the example
//     shown above, this is specified in the "fcst" dictionary and copied to
//     "obs".
//
//   - Simplified vertical level matching logic is applied for surface message
//     types. Observations for the following message types are assumed to be at
//     the surface, as defined by the default message_type_group_map:
//        ADPSFC, SFCSHP, MSONET
//
//   - The "message_type" would be placed in the "field" array element if more
//     than one "message_type" entry is desired within the config file.
```

```
//     e.g.
//     fcst = {
//       censor_thresh = [];
//       censor_val    = [];
//       cnt_thresh    = [ NA ];
//       cnt_logic     = UNION;
//       wind_thresh   = [ NA ];
//       wind_logic    = UNION;
//
//       field = [
//          {
//            message_type = [ "ADPUPA" ];
//            sid_inc      = [];
//            sid_exc      = [];
//            name         = "TMP";
//            level        = [ "P250", "P500", "P700", "P850", "P1000" ];
//            cat_thresh   = [ <=273.0 ];
//          },
//          {
//            message_type = [ "ADPSFC" ];
//            sid_inc      = [];
//            sid_exc      = [ "KDEN", "KDET" ];
//            name         = "TMP";
//            level        = [ "Z2" ];
//            cat_thresh   = [ <=273.0 ];
//          }
//       ];
//     }
//
//    - The "sid_inc" entry is an array of station ID groups indicating which
//      station ID's should be included in the verification task. If specified,
//      only those station ID's appearing in the list will be included.  Note
//      that filtering by station ID may also be accomplished using the "mask.sid"
//      option. However, when using the "sid_inc" option, statistics are reported
//      separately for each masking region.
//    - The "sid_exc" entry is an array of station ID groups indicating which
//      station ID's should be excluded from the verification task.
//    - Each element in the "sid_inc" and "sid_exc" arrays is either the name of
//      a single station ID or the full path to a station ID group file name.
//      A station ID group file consists of a name for the group followed by a
//      list of station ID's. All of the station ID's indicated will be concatenated
//      into one long list of station ID's to be included or excluded.
//    - As with "message_type" above, the "sid_inc" and "sid_exc" settings can be
```

```
//      placed in the in the "field" array element to control which station ID's
//      are included or excluded for each verification task.
//
obs = fcst;

//
// The "climo_mean" dictionary specifies climatology mean data to be read by the
// Grid-Stat, Point-Stat, Ensemble-Stat, and Series-Analysis tools. It consists
// of several entires defining the climatology file names and fields to be used.
//
//   - The "file_names" entry specifies one or more file names containing
//     the gridded climatology data to be used.
//
//   - The "field" entry is an array of dictionaries, specified the same
//     way as those in the "fcst" and "obs" dictionaries. If the array has
//     length zero, not climatology data will be read and all climatology
//     statistics will be written as missing data. Otherwise, the array
//     length must match the length of "field" in the "fcst" and "obs"
//     dictionaries.
//
//   - The "regrid" dictionary defines how the climatology data should be
//     regridded to the verification domain.
//
//   - The "time_interp_method" entry specifies how the climatology data should
//     be interpolated in time to the forecast valid time:
//      - NEAREST for data closest in time
//      - UW_MEAN for average of data before and after
//      - DW_MEAN for linear interpolation in time of data before and after
//
//   - The "day_interval" entry is an integer specifying the spacing in days of
//     the climatology data. Use 31 for monthly data or 1 for daily data.
//     Use "NA" if the timing of the climatology data should not be checked.
//
//   - The "hour_interval" entry is an integer specifying the spacing in hours of
//     the climatology data for each day. This should be set between 0 and 24,
//     with 6 and 12 being common choices. Use "NA" if the timing of the
//     climatology data should not be checked.
//
//   - The "day_interval" and "hour_interval" entries replace the deprecated
//     entries "match_month", "match_day", and "time_step".
//
climo_mean = {
```

```
   file_name = [ "/path/to/climatological/mean/files" ];
   field     = [];

   regrid = {
      method     = NEAREST;
      width      = 1;
      vld_thresh = 0.5;
   }

   time_interp_method = DW_MEAN;
   day_interval       = 31;
   hour_interal       = 6;
}

//
// The "climo_stdev" dictionary specifies climatology standard deviation data to
// be read by the Grid-Stat, Point-Stat, Ensemble-Stat, and Series-Analysis
// tools. The "climo_mean" and "climo_stdev" data define the climatological
// distribution for each grid point, assuming normality. These climatological
// distributions are used in two ways:
// (1) To define climatological distribution percentile (CDP) thresholds which
//     can be used as categorical (cat_thresh), continuous (cnt_thresh), or wind
//     speed (wind_thresh) thresholds.
// (2) To subset matched pairs into climatological bins based on where the
//     observation value falls within the climatological distribution. See the
//     "climo_cdf" dictionary.
//
// This dictionary is identical to the "climo_mean" dictionary described above
// but points to files containing climatological standard deviation values
// rather than means. In the example below, this dictionary is set by copying
// over the "climo_mean" setting and then updating the "file_name" entry.
//
climo_stdev = climo_mean;
climo_stdev = {
   file_name = [ "/path/to/climatological/standard/deviation/files" ];
}

//
// The "climo_cdf" dictionary specifies how the the climatological mean
// ("climo_mean") and standard deviation ("climo_stdev") data are used to
// evaluate model performance relative to where the observation value falls
// within the climatological distribution. This dictionary consists of 3
// entries:
```

```
// (1) The "cdf_bins" entry defines the climatological bins either as an integer
//     or an array of floats between 0 and 1.
// (2) The "center_bins" entry may be set to TRUE or FALSE.
// (3) The "write_bins" entry may be set to TRUE or FALSE.
//
// MET uses the climatological mean and standard deviation to construct a normal
// PDF at each observation location. The total area under the PDF is 1, and the
// climatological CDF value is computed as the area of the PDF to the left of
// the observation value. Since the CDF is a value between 0 and 1, the CDF
// bins must span that same range.
//
// When "cdf_bins" is set to an array of floats, they explicitly define the
// climatological bins. The array must begin with 0.0 and end with 1.0.
// For example:
//   cdf_bins = [ 0.0, 0.10, 0.25, 0.75, 0.90, 1.0 ];
//
// When "cdf_bins" is set to an integer, it defines the number of bins to be
// used. The "center_bins" flag indicates whether or not the bins should be
// centered on 0.5. An odd number of bins can be centered or uncentered while
// an even number of bins can only be  uncentered. For example:
//   4 uncentered bins (cdf_bins = 4; center_bins = FALSE;) yields:
//     0.0, 0.25, 0.50, 0.75, 1.0
//   5 uncentered bins (cdf_bins = 5; center_bins = FALSE;) yields:
//     0.0, 0.2, 0.4, 0.6, 0.8, 0.9, 1.0
//   5   centered bins (cdf_bins = 5; center_bins = TRUE;) yields:
//     0.0, 0.125, 0.375, 0.625, 0.875, 1.0
//
// When multiple climatological bins are used, statistics are computed
// separately for each bin, and the average of the statistics across those bins
// is written to the output. When "write_bins" is true, the statistics for each
// bin are also written to the output. The bin number is appended to the
// contents of the VX_MASK output column.
//
// Setting the number of bins to 1 effectively disables this logic by grouping
// all pairs into a single climatological bin.
//
climo_cdf = {
   cdf_bins    = 11;    // or an array of floats
   center_bins = TRUE;  // or FALSE
   write_bins  = FALSE; // or TRUE
}

//
```

```
// When specifying climatology data for probability forecasts, either supply a
// probabilistic "climo_mean" field or non-probabilistic "climo_mean" and
// "climo_stdev" fields from which a normal approximation of the climatological
// probabilities should be derived.
//
// When "climo_mean" is set to a probability field with a range of [0, 1] and
// "climo_stdev" is unset, the MET tools use the "climo_mean" probability values
// directly to compute Brier Skill Score (BSS).
//
// When "climo_mean" and "climo_stdev" are both set to non-probability fields,
// the MET tools use the mean, standard deviation, and observation event
// threshold to derive a normal approximation of the climatological
// probabilities. Those derived probability values are used to compute BSS.
//

// The "mask_missing_flag" entry specifies how missing data should be handled
// in the Wavelet-Stat and MODE tools:
//    - "NONE" to perform no masking of missing data
//    - "FCST" to mask the forecast field with missing observation data
//    - "OBS" to mask the observation field with missing forecast data
//    - "BOTH" to mask both fields with missing data from the other
//
mask_missing_flag = BOTH;

//
// The "obs_window" entry is a dictionary specifying a beginning ("beg"
// entry) and ending ("end" entry) time offset values in seconds. It defines
// the time window over which observations are retained for scoring. These time
// offsets are defined relative to a reference time t, as [t+beg, t+end].
// In PB2NC, the reference time is the PREPBUFR files center time. In
// Point-Stat and Ensemble-Stat, the reference time is the forecast valid time.
//
obs_window = {
   beg = -5400;
   end =  5400;
}

//
// The "mask" entry is a dictionary that specifies the verification masking
// regions to be used when computing statistics. Each mask defines a
// geographic extent, and any matched pairs falling inside that area will be
// used in the computation of statistics. Masking regions may be specified
// in the following ways:
```

```
//
//    - The "grid" entry is an array of named grids. It contains a
//      comma-separated list of pre-defined NCEP grids over which to perform
//      verification. An empty list indicates that no masking grids should be
//      used. The standard NCEP grids are named "GNNN" where NNN indicates the
//      three digit grid number. Supplying a value of "FULL" indicates that the
//      verification should be performed over the entire grid on which the data
//      resides.
//      http://www.nco.ncep.noaa.gov/pmb/docs/on388/tableb.html
//      The "grid" entry can be the gridded data file defining grid.
//
//    - The "poly" entry contains a comma-separated list of files that define
//      verification masking regions. These masking regions may be specified in
//      two ways: as a lat/lon polygon or using a gridded data file such as the
//      NetCDF output of the Gen-Vx-Mask tool.
//
//       - An ASCII file containing a lat/lon polygon.
//         Latitude in degrees north and longitude in degrees east.
//         The first and last polygon points are connected.
//         e.g. "MET_BASE/poly/EAST.poly" which consists of n points:
//              "poly_name lat1 lon1 lat2 lon2... latn lonn"
//
//         Several masking polygons used by NCEP are predefined in the
//         installed share/met/poly directory. Creating a new polygon is as
//         simple as creating a text file with a name for the polygon followed
//         by the lat/lon points which define its boundary. Adding a new masking
//         polygon requires no code changes and no recompiling. Internally, the
//         lat/lon polygon points are converted into x/y values in the grid. The
//         lat/lon values for the observation points are also converted into x/y
//         grid coordinates. The computations performed to check whether the
//         observation point falls within the polygon defined is done in x/y
//         grid space.
//
//       - The NetCDF output of the gen_vx_mask tool.
//
//       - Any gridded data file that MET can read may be used to define a
//         verification masking region. Users must specify a description of the
//         field to be used from the input file and, optionally, may specify a
//         threshold to be applied to that field. Once this threshold is
//         applied, any grid point where the resulting field is 0, the mask is
//         turned off. Any grid point where it is non-zero, the mask is turned
//         on.
//         e.g. "sample.grib {name = \"TMP\"; level = \"Z2\";} >273"
```

```
//
//    - The "sid" entry is an array of strings which define groups of
//      observation station ID's over which to compute statistics. Each entry
//      in the array is either a filename of a comma-separated list.
//       - For a filename, the strings are whitespace-separated. The first
//         string is the mask "name" and the remaining strings are the station
//         ID's to be used.
//       - For a comma-separated list, optionally use a colon to specify a name.
//         For "MY_LIST:SID1,SID2", name = MY_LIST and values = SID1 and SID2.
//       - For a comma-separated list of length one with no name specified, the
//         mask "name" and value are both set to the single station ID string.
//         For "SID1", name = SID1 and value = SID1.
//       - For a comma-separated list of length greater than one with no name
//         specified, the name is set to MASK_SID and the values are the station
//         ID's to be used.
//         For "SID1,SID2", name = MASK_SID and values = SID1 and SID2.
//       - The "name" of the station ID mask is written to the VX_MASK column
//         of the MET output files.
//    - The "llpnt" entry is either a single dictionary or an array of
//      dictionaries. Each dictionary contains three entries, the "name" for
//      the masking region, "lat_thresh", and "lon_thresh". The latitude and
//      longitude thresholds are applied directly to the point observation
//      latitude and longitude values. Only observations whose latitude and
//      longitude values meet this threshold criteria are used. A threshold set
//      to "NA" always evaluates to true.
//
// The masking logic for processing point observations in Point-Stat and
// Ensemble-Stat fall into two cateogries. The "sid" and "llpnt" options apply
// directly to the point observations. Only those observations for the specified
// station id's are included in the "sid" masks. Only those observations meeting
// the latitude and longitude threshold criteria are included in the "llpnt"
// masks.
//
// The "grid" and "poly" mask options are applied to the grid points of the
// verification domain. Each grid point is determined to be inside or outside
// the masking region. When processing point observations, their latitude and
// longitude values are rounded to the nearest grid point of the verification
// domain. If the nearest grid point is inside the mask, that point observation
// is included in the mask.
//
mask = {
   grid    = [ "FULL" ];
   poly    = [ "MET_BASE/poly/LMV.poly",
```

```
              "MET_BASE/out/gen_vx_mask/CONUS_poly.nc",
              "MET_BASE/sample_fcst/2005080700/wrfprs_ruc13_12.tm00_G212 \
              {name = \"TMP\"; level = \"Z2\";} >273"
            ];
   sid     = [ "CONUS.stations" ];
   llpnt   = [ { name       = "LAT30TO40";
                 lat_thresh = >=30&&<=40;
                 lon_thresh = NA; },
               { name       = "BOX";
                 lat_thresh = >=20&&<=40;
                 lon_thresh = >=-110&&<=-90; } ];
}

//
// The "ci_alpha" entry is an array of floats specifying the values for alpha
// to be used when computing confidence intervals. Values of alpha must be
// between 0 and 1. The confidence interval computed is 1 minus the alpha
// value. Therefore, an alpha value of 0.05 corresponds to a 95% confidence
// interval.
//
ci_alpha = [ 0.05, 0.10 ];

//
// The "boot" entry defines the parameters to be used in calculation of
// bootstrap confidence intervals. The interval variable indicates what method
// should be used for computing bootstrap confidence intervals:
//
//    - The "interval" entry specifies the confidence interval method:
//       - "BCA" for the BCa (bias-corrected percentile) interval method is
//         highly accurate but computationally intensive.
//       - "PCTILE" uses the percentile method which is somewhat less accurate
//         but more efficient.
//
//    - The "rep_prop" entry specifies a proportion between 0 and 1 to define
//      the replicate sample size to be used when computing percentile
//      intervals. The replicate sample size is set to boot_rep_prop * n,
//      where n is the number of raw data points.
//
//      When computing bootstrap confidence intervals over n sets of matched
//      pairs, the size of the subsample, m, may be chosen less than or equal to
//      the size of the sample, n. This variable defines the size of m as a
//      proportion relative to the size of n. A value of 1 indicates that the
//      size of the subsample, m, should be equal to the size of the sample, n.
```

```
//
//    - The "n_rep" entry defines the number of subsamples that should be taken
//      when computing bootstrap confidence intervals. This variable should be
//      set large enough so that when confidence intervals are computed multiple
//      times for the same set of data, the intervals do not change much.
//      Setting this variable to zero disables the computation of bootstrap
//      confidence intervals, which may be necessary to run MET in realtime or
//      near-realtime over large domains since bootstrapping is computationally
//      expensive. Setting this variable to 1000 indicates that bootstrap
//      confidence interval should be computed over 1000 subsamples of the
//      matched pairs.
//
//    - The "rng" entry defines the random number generator to be used in the
//      computation of bootstrap confidence intervals. Subsamples are chosen at
//      random from the full set of matched pairs. The randomness is determined
//      by the random number generator specified. Users should refer to detailed
//      documentation of the GNU Scientific Library for a listing of the random
//      number generators available for use.
//      http://www.gnu.org/software/gsl/manual/html_node/
//      Random-Number-Generator-Performance.html
//
//    - The "seed" entry may be set to a specific value to make the computation
//      of bootstrap confidence intervals fully repeatable. When left empty
//      the random number generator seed is chosen automatically which will lead
//      to slightly different bootstrap confidence intervals being computed each
//      time the data is run. Specifying a value here ensures that the bootstrap
//      confidence intervals will be reproducable over multiple runs on the same
//      computing platform.
//
boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
}

//
// The "interp" entry is a dictionary that specifies what interpolation or
// smoothing (for the Grid-Stat tool) methods should be applied.
// This dictionary may include the following entries:
//
//    - The "field" entry specifies to which field(s) the interpolation method
```

```
//      should be applied. This does not apply when doing point verification
//      with the Point-Stat or Ensemble-Stat tools:
//       - "FCST" to interpolate/smooth the forecast field.
//       - "OBS" to interpolate/smooth the observation field.
//       - "BOTH" to interpolate/smooth both the forecast and the observation.
//
//    - The "vld_thresh" entry specifies a number between 0 and 1. When
//      performing interpolation over some neighborhood of points the ratio of
//      the number of valid data points to the total number of points in the
//      neighborhood is computed. If that ratio is less than this threshold,
//      the matched pair is discarded. Setting this threshold to 1, which is the
//      default, requires that the entire neighborhood must contain valid data.
//      This variable will typically come into play only along the boundaries of
//      the verification region chosen.
//
//    - The "shape" entry may be set to SQUARE or CIRCLE to specify the shape
//      of the smoothing area.
//
//    - The "type" entry is an array of dictionaries, each specifying an
//      interpolation method. Interpolation is performed over a N by N box
//      centered on each point, where N is the width specified. Each of these
//      dictionaries must include:
//
//      - The "width" entry is an integer which specifies the size of the
//        interpolation area. The area is either a square or circle containing
//        the observation point. The width value specifies the width of the
//        square or diameter of the circle. A width value of 1 is interpreted
//        as the nearest neighbor model grid point to the observation point.
//        For squares, a width of 2 defines a 2 x 2 box of grid points around
//        the observation point (the 4 closest model grid points), while a width
//        of 3 defines a 3 x 3 box of grid points around the observation point,
//        and so on. For odd widths in grid-to-point comparisons
//        (i.e. Point-Stat), the interpolation area is centered on the model
//        grid point closest to the observation point. For grid-to-grid
//        comparisons (i.e. Grid-Stat), the width must be odd.
//
//      - The "method" entry specifies the interpolation procedure to be
//        applied to the points in the box:
//         - MIN         for the minimum value
//         - MAX         for the maximum value
//         - MEDIAN      for the median value
//         - UW_MEAN     for the unweighted average value
//         - DW_MEAN     for the distance-weighted average value
```

```
//                          where weight = distance^-2
//         - LS_FIT      for a least-squares fit
//         - BILIN       for bilinear interpolation (width = 2)
//         - NEAREST     for the nearest grid point (width = 1)
//         - BEST        for the value closest to the observation
//         - UPPER_LEFT  for the upper left grid point (width = 1)
//         - UPPER_RIGHT for the upper right grid point (width = 1)
//         - LOWER_RIGHT for the lower right grid point (width = 1)
//         - LOWER_LEFT  for the lower left grid point (width = 1)
//         - GAUSSIAN    for the Gaussian kernel
//         - MAXGAUSS    for the maximum value followed by a Gaussian smoother
//         - GEOG_MATCH  for the nearest grid point where the land/sea mask
//                          and geography criteria are satisfied.
//
//         The BUDGET, FORCE, GAUSSIAN, and MAXGAUSS methods are not valid for
//         interpolating to point locations. For grid-to-grid comparisons, the
//         only valid smoothing methods are MIN, MAX, MEDIAN, UW_MEAN, and
//         GAUSSIAN, and MAXGAUSS.
//
interp = {
   field      = BOTH;
   vld_thresh = 1.0;
   shape      = SQUARE;

   type = [
      {
         method = UW_MEAN;
         width  = 1;
      }
   ];
}

//
// The "nbrhd" entry is a dictionary that is very similar to the "interp"
// entry. It specifies information for computing neighborhood statistics in
// Grid-Stat. This dictionary may include the following entries:
//
//    - The "field" entry specifies to which field(s) the computation of
//      fractional coverage should be applied. Grid-Stat processes each
//      combination of categorical threshold and neighborhood width to
//      derive the fractional coverage fields from which neighborhood
//      statistics are calculated. Users who have computed fractional
//      coverage fields outside of MET can use this option to disable
```

```
//      these computations. Instead, the raw input values will be
//      used directly to compute neighborhood statistics:
//       - "BOTH" to compute fractional coverage for both the forecast
//                and the observation fields (default).
//       - "FCST" to only process the forecast field.
//       - "OBS"  to only process the observation field.
//       - "NONE" to process neither field.
//
//    - The "vld_thresh" entry is described above.
//
//    - The "shape" entry defines the shape of the neighborhood.
//      Valid values are "SQUARE" or "CIRCLE"
//
//    - The "width" entry is as described above, and must be odd.
//
//    - The "cov_thresh" entry is an array of thresholds to be used when
//      computing categorical statistics for the neighborhood fractional
//      coverage field.
//
nbrhd = {
   field      = BOTH;
   vld_thresh = 1.0;
   shape      = SQUARE;
   width      = [ 1 ];
   cov_thresh = [ >=0.5 ];
}

//
// The "fourier" entry is a dictionary which specifies the application of the
// Fourier decomposition method. It consists of two arrays of the same length
// which define the beginning and ending wave numbers to be included. If the
// arrays have length zero, no Fourier decomposition is applied. For each array
// entry, the requested Fourier decomposition is applied to the forecast and
// observation fields. The beginning and ending wave numbers are indicated in
// the MET ASCII output files by the INTERP_MTHD column (e.g. WV1_0-3 for waves
// 0 to 3 or WV1_10 for only wave 10). This 1-dimensional Fourier decomposition
// is computed along the Y-dimension only (i.e. the columns of data). It is only
// defined when each grid point contains valid data. If either input field
// contains missing data, no Fourier decomposition is computed.
//
// The available wave numbers start at 0 (the mean across each row of data)
// and end at (Nx+1)/2 (the finest level of detail), where Nx is the X-dimension
// of the verification grid:
```

```
//
//    - The "wave_1d_beg" entry is an array of integers specifying the first
//      wave number to be included.
//
//    - The "wave_1d_end" entry is an array of integers specifying the last
//      wave number to be included.
//
fourier = {
   wave_1d_beg = [ 0, 4, 10 ];
   wave_1d_end = [ 3, 9, 20 ];
}

//
// The "gradient" entry is a dictionary which specifies the number and size of
// gradients to be computed. The "dx" and "dy" entries specify the size of the
// gradients in grid units in the X and Y dimensions, respectively. dx and dy
// are arrays of integers (positive or negative) which must have the same
// length, and the GRAD output line type will be computed separately for each
// entry. When computing gradients, the value at the (x, y) grid point is
// replaced by the value at the (x+dx, y+dy) grid point minus the value at
// (x, y).
//
// This configuration option may be set separately in each "obs.field" entry.
//
gradient = {
   dx = [ 1 ];
   dy = [ 1 ];
}

//
// The "distance_map" entry is a dictionary containing options related to the
// distance map statistics in the DMAP output line type. The "baddeley_p" entry
// is an integer specifying the exponent used in the Lp-norm when computing the
// Baddeley Delta metric. The "baddeley_max_dist" entry is a floating point
// number specifying the maximum allowable distance for each distance map. Any
// distances larger than this number will be reset to this constant. A value of
// NA indicates that no maximum distance value should be used. The "fom_alpha"
// entry is a floating point number specifying the scaling constant to be used
// when computing Pratt's Figure of Merit. The "zhu_weight" specifies a value
// between 0 and 1 to define the importance of the RMSE of the binary fields
// (i.e. amount of overlap) versus the mean-error distance (MED). The default
// value of 0.5 gives equal weighting.
//
```

```
// This configuration option may be set separately in each "obs.field" entry.
//
distance_map = {
   baddeley_p        = 2;
   baddeley_max_dist = NA;
   fom_alpha         = 0.1;
   zhu_weight        = 0.5;
}

//
// The "land_mask" dictionary defines the land/sea mask field which is used
// when verifying at the surface. For point observations whose message type
// appears in the "LANDSF" entry of the "message_type_group_map" setting,
// only use forecast grid points where land = TRUE. For point observations
// whose message type appears in the "WATERSF" entry of the
// "message_type_group_map" setting, only use forecast grid points where
// land = FALSE. The "flag" entry enables/disables this logic. If the
// "file_name" entry is left empty, then the land/sea is assumed to exist in
// the input forecast file. Otherwise, the specified file(s) are searched for
// the data specified in the "field" entry. The "regrid" settings specify how
// this field should be regridded to the verification domain. Lastly, the
// "thresh" entry is the threshold which defines land (threshold is true) and
// water (threshold is false).
// land_mask.flag may be set separately in each "obs.field" entry.
//
land_mask = {
   flag      = FALSE;
   file_name = [];
   field     = { name = "LAND"; level = "L0"; }
   regrid    = { method = NEAREST; width = 1; }
   thresh    = eq1;
}

//
// The "topo_mask" dictionary defines the model topography field which is used
// when verifying at the surface. This logic is applied to point observations
// whose message type appears in the "SURFACE" entry of the
// "message_type_group_map" setting. Only use point observations where the
// topo - station elevation difference meets the "use_obs_thresh" threshold
// entry. For the observations kept, when interpolating forecast data to the
// observation location, only use forecast grid points where the topo - station
// difference meets the "interp_fcst_thresh" threshold entry. The flag entry
// enables/disables this logic. If the "file_name" is left empty, then the
```

```
// topography data is assumed to exist in the input forecast file. Otherwise,
// the specified file(s) are searched for the data specified in the "field"
// entry. The "regrid" settings specify how this field should be regridded to
// the verification domain.
// topo_mask.flag may be set separately in each "obs.field" entry.
//
topo_mask = {
   flag               = FALSE;
   file_name          = [];
   field              = { name = "TOPO"; level = "L0"; }
   regrid             = { method = BILIN; width = 2; }
   use_obs_thresh     = ge-100&&le100;
   interp_fcst_thresh = ge-50&&le50;
}

//
// The "hira" entry is a dictionary that is very similar to the "interp" and
// "nbrhd" entries. It specifies information for applying the High Resolution
// Assessment (HiRA) verification logic in Point-Stat. HiRA is analogous to
// neighborhood verification but for point observations. The HiRA logic
// interprets the forecast values surrounding each point observation as an
// ensemble forecast. These ensemble values are processed in two ways. First,
// the ensemble continuous statistics (ECNT) and ranked probability score (RPS)
// line types are computed directly from the ensemble values. Second, for each
// categorical threshold specified, a fractional coverage value is computed as
// the ratio of the nearby forecast values that meet the threshold criteria.
// Point-Stat evaluates those fractional coverage values as if they were a
// probability forecast. When applying HiRA, users should enable the matched
// pair (MPR), probabilistic (PCT, PSTD, PJC, or PRC), or ensemble statistics
// (ECNT or PRS) line types in the output_flag dictionary. The number of
// probabilistic HiRA output lines is determined by the number of categorical
// forecast thresholds and HiRA neighborhood widths chosen.
// This dictionary may include the following entries:
//
//    - The "flag" entry is a boolean which toggles "hira"
//      on (TRUE) and off (FALSE).
//
//    - The "width" entry specifies the neighborhood size. Since HiRA applies
//      to point observations, the width may be even or odd.
//
//    - The "vld_thresh" entry is as described above.
//
//    - The "cov_thresh" entry is an array of probabilistic thresholds used to
```

```
//      populate the Nx2 probabilistic contingency table written to the PCT
//      output line and used for computing probabilistic statistics.
//
//    - The "shape" entry defines the shape of the neighborhood.
//      Valid values are "SQUARE" or "CIRCLE"
//
//    - The "prob_cat_thresh" entry defines the thresholds which define ensemble
//      probabilities from which to compute the ranked probability score output.
//      If left empty but climatology data is provided, the climo_cdf thresholds
//      will be used instead.
//
hira = {
   flag            = FALSE;
   width           = [ 2, 3, 4, 5 ];
   vld_thresh      = 1.0;
   cov_thresh      = [ ==0.25 ];
   shape           = SQUARE;
   prob_cat_thresh = [];
}

//
// The "output_flag" entry is a dictionary that specifies what verification
// methods should be applied to the input data. Options exist for each
// output line type from the MET tools. Each line type may be set to one of:
//    - "NONE" to skip the corresponding verification method
//    - "STAT" to write the verification output only to the ".stat" output file
//    - "BOTH" to write to the ".stat" output file as well the optional
//      "_type.txt" file, a more readable ASCII file sorted by line type.
//
output_flag = {
   fho    = NONE;  // Forecast, Hit, Observation Rates
   ctc    = NONE;  // Contingency Table Counts
   cts    = NONE;  // Contingency Table Statistics
   mctc   = NONE;  // Multi-category Contingency Table Counts
   mcts   = NONE;  // Multi-category Contingency Table Statistics
   cnt    = NONE;  // Continuous Statistics
   sl1l2  = NONE;  // Scalar L1L2 Partial Sums
   sal1l2 = NONE;  // Scalar Anomaly L1L2 Partial Sums when climatological data
                   // is supplied
   vl1l2  = NONE;  // Vector L1L2 Partial Sums
   val1l2 = NONE;  // Vector Anomaly L1L2 Partial Sums when climatological data
                   // is supplied
   pct    = NONE;  // Contingency Table Counts for Probabilistic Forecasts
```

```
   pstd   = NONE;  // Contingency Table Statistics for Probabilistic Forecasts
                   // with Dichotomous outcomes
   pjc    = NONE;  // Joint and Conditional Factorization for Probabilistic
                   // Forecasts
   prc    = NONE;  // Receiver Operating Characteristic for Probabilistic
                   // Forecasts
   eclv   = NONE;  // Economic Cost/Loss Value derived from CTC and PCT lines
   mpr    = NONE;  // Matched Pair Data
   nbrctc = NONE;  // Neighborhood Contingency Table Counts
   nbrcts = NONE;  // Neighborhood Contingency Table Statistics
   nbrcnt = NONE;  // Neighborhood Continuous Statistics
   isc    = NONE;  // Intensity-Scale
   ecnt   = NONE;  // Ensemble Continuous Statistics
   rps    = NONE;  // Ranked Probability Score Statistics
   rhist  = NONE;  // Rank Histogram
   phist  = NONE;  // Probability Integral Transform Histogram
   orank  = NONE;  // Observation Rank
   ssvar  = NONE;  // Spread Skill Variance
   grad   = NONE;  // Gradient statistics (S1 score)
}

//
// The "nc_pairs_flag" can be set either to a boolean value or a dictionary
// in either Grid-Stat, Wavelet-Stat or MODE. The dictionary (with slightly
// different entries for the various tools ... see the default config files)
// has individual boolean settings turning on or off the writing out of the
// various fields in the netcdf output file for the tool. Setting all
// dictionary entries to false means the netcdf file will not be generated.
//
// "nc_pairs_flag" can also be set to a boolean value. In this case, a value
// of true means to just accept the default settings (which will turn on
// the output of all the different fields). A value of false means no
// netcdf output will be generated.
//
nc_pairs_flag = {
   latlon       = TRUE;
   raw          = TRUE;
   diff         = TRUE;
   climo        = TRUE;
   climo_cdp    = FALSE;
   weight       = FALSE;
   nbrhd        = FALSE;
   fourier      = FALSE;
```

```
   gradient     = FALSE;
   distance_map = FLASE;
   apply_mask   = TRUE;
}

//
// The "nc_pairs_var_name" entry specifies a string for each verification task
// in Grid-Stat. This string is parsed from each "obs.field" dictionary entry
// and is used to construct variable names for the NetCDF matched pairs output
// file. The default value of an empty string indicates that the "name" and
// "level" strings of the input data should be used.  If the input data "level"
// string changes for each run of Grid-Stat, using this option to define a
// constant string may make downstream processing more convenient.
//
// e.g. nc_pairs_var_name = "TMP";
//
nc_pairs_var_name = "";

//
// The "nc_pairs_var_suffix" entry is similar to the "nc_pairs_var_name" entry
// described above.  It is also parsed from each "obs.field" dictionary entry.
// However, it defines a suffix to be appended to the output variable name.
// This enables the output variable names to be made unique. For example, when
// verifying height for multiple level types but all with the same level value,
// use this option to customize the output variable names.
//
// e.g. nc_pairs_var_suffix = "TROPO"; (for the tropopause height)
//      nc_pairs_var_suffix = "FREEZING"; (for the freezing level height)
//
// NOTE: This option was previously named "nc_pairs_var_str", which is
//       now deprecated.
//
nc_pairs_var_suffix = "";

//
// The "ps_plot_flag" entry is a boolean value for Wavelet-Stat and MODE
// indicating whether a PostScript plot should be generated summarizing
// the verification.
//
ps_plot_flag = TRUE;

//
// The "grid_weight_flag" specifies how grid weighting should be applied
```

```
// during the computation of continuous statistics and partial sums. It is
// meant to account for grid box area distortion and is often applied to global
// Lat/Lon grids. It is only applied for grid-to-grid verification in Grid-Stat
// and Ensemble-Stat and is not applied for grid-to-point verification.
// Three grid weighting options are currently supported:
//
//    - "NONE" to disable grid weighting using a constant weight (default).
//    - "COS_LAT" to define the weight as the cosine of the grid point latitude.
//      This an approximation for grid box area used by NCEP and WMO.
//    - "AREA" to define the weight as the true area of the grid box (km^2).
//
// The weights are ultimately computed as the weight at each grid point divided
// by the sum of the weights for the current masking region.
//
grid_weight_flag = NONE;

//
// The "rank_corr_flag" entry is a boolean to indicate whether Kendall's Tau
// and Spearman's Rank Correlation Coefficients (in the CNT line type) should
// be computed. Computing them over large datasets is computationally
// intensive and slows down the runtime significantly.
//
rank_corr_flag = FALSE;

//
// The "duplicate_flag" entry specifies how to handle duplicate point
// observations in Point-Stat and Ensemble-Stat:
//
//    - "NONE" to use all point observations (legacy behavior)
//    - "UNIQUE" only use a single observation if two or more observations
//      match. Matching observations are determined if they contain identical
//      latitude, longitude, level, elevation, and time information.
//      They may contain different observation values or station IDs
//
// The reporting mechanism for this feature can be activated by specifying
// a verbosity level of three or higher. The report will show information
// about where duplicates were detected and which observations were used
// in those cases.
//
duplicate_flag = NONE;

//
// The "obs_summary" entry specifies how to compute statistics on
```

```
// observations that appear at a single location (lat,lon,level,elev)
// in Point-Stat and Ensemble-Stat. Eight techniques are
// currently supported:
//
//    - "NONE" to use all point observations (legacy behavior)
//    - "NEAREST" use only the observation that has the valid
//      time closest to the forecast valid time
//    - "MIN" use only the observation that has the lowest value
//    - "MAX" use only the observation that has the highest value
//    - "UW_MEAN" compute an unweighted mean of the observations
//    - "DW_MEAN" compute a weighted mean of the observations based
//      on the time of the observation
//    - "MEDIAN" use the median observation
//    - "PERC" use the Nth percentile observation where N = obs_perc_value
//
// The reporting mechanism for this feature can be activated by specifying
// a verbosity level of three or higher. The report will show information
// about where duplicates were detected and which observations were used
// in those cases.
//
obs_summary = NONE;

//
// Percentile value to use when obs_summary = PERC
//
obs_perc_value = 50;

//
// The "obs_quality" entry specifies the quality flag values that are to be
// retained and used for verification. An empty list signifies that all
// point observations should be used, regardless of their quality flag value.
// The quality flag values will vary depending on the original source of the
// observations. The quality flag values to retain should be specified as
// an array of strings, even if the values themselves are numeric.
//
obs_quality = [ "1", "2", "3", "9" ];

//
// The "met_data_dir" entry specifies the location of the internal MET data
// sub-directory which contains data files used when generating plots. It
// should be set to the installed share/met directory so the MET tools can
// locate the static data files they need at run time.
//
```

```
met_data_dir = "MET_BASE";

//
// The "fcst_raw_plot" entry is a dictionary used by Wavelet-Stat and MODE
// containing colortable plotting information for the plotting of the raw
// forecast field:
//
//    - The "color_table" entry specifies the location and name of the
//      colortable file to be used.
//
//    - The "plot_min" and "plot_max" entries specify the range of data values.
//      If they are both set to 0, the MET tools will automatically rescale
//      the colortable to the range of values present in the data. If they
//      are not both set to 0, the MET tools will rescale the colortable using
//      their values.
//
fcst_raw_plot = {
   color_table = "MET_BASE/colortables/met_default.ctable";
   plot_min = 0.0;
   plot_max = 0.0;
}

//
// The "obs_raw_plot", "wvlt_plot", and "object_plot" entries are dictionaries
// similar to the "fcst_raw_plot" described above.
//

//
// The "tmp_dir" entry is a string specifying the location where temporary
// files should be written.
//
tmp_dir = "/tmp";

//
// The "output_prefix" entry specifies a string to be included in the output
// file name. The MET statistics tools construct output file names that
// include the tool name and timing information. You can use this setting
// to modify the output file name and avoid naming conflicts for multiple runs
// of the same tool.
//
output_prefix  = "";

//
```

```
// The "version" entry specifies the version number of the configuration file.
// The configuration file version number should match the version number of
// the MET code being run. This value should generally not be modified.
//
version = "V6.0";

//
// This feature was implemented to allow additional processing of observations
// with high temporal resolution. The "flag" entry toggles the "time_summary"
// on (TRUE) and off (FALSE). Obs may be summarized across the user specified
// time period defined by the "beg" and "end" entries. The "step" entry defines
// the time between intervals in seconds. The "width" entry specifies the
// summary interval in seconds. It may either be set as an integer number of
// seconds for a centered time interval or a dictionary with beginning and
// ending time offsets in seconds.
//
// e.g. beg = "00";
//      end = "235959";
//      step = 300;
//      width = 600;
//      width = { beg = -300; end = 300; }
//
// This example does a 10-minute time summary every 5 minutes throughout the
// day. The first interval will be from 23:55:00 the previous day through
// 00:04:59 of the current day. The second interval will be from 0:00:00
// through 00:09:59. And so on.
//
// The two "width" settings listed above are equivalent. Both define a centered
// 10-minute time interval. Use the "beg" and "end" entries to define
// uncentered time intervals. The following example requests observations for
// one hour prior:
//      width = { beg = -3600; end = 0; }
//
// The summaries will only be calculated for the specified GRIB codes.
// The supported summaries are "min" (minimum), "max" (maximum), "range",
// "mean", "stdev" (standard deviation), "median" and "p##" (percentile, with
// the desired percentile value specified in place of ##).
//
// The "vld_freq" and "vld_thresh" options may be used to require that a certain
// ratio of observations must be present and contain valid data within the time
// window in order for a summary value to be computed. The "vld_freq" entry
// defines the expected observation frequency in seconds. For example, when
// summarizing 1-minute data (vld_freq = 60) over a 30 minute time window,
```

```
// setting "vld_thresh = 0.5" requires that at least 15 of the 30 expected
// observations be present and valid for a summary value to be written. The
// default "vld_thresh = 0.0" setting will skip over this logic.
//
// The variable names are saved to NetCDF file if they are given instead of
// grib_codes which are not available for non GRIB input. The "obs_var" option
// was added and works like "grib_code" option (string value VS. int value).
// They are inclusive (union). All variables are included if both options
// are empty. Note: grib_code 11 is equivalent to obs_var "TMP".
//
time_summary = {
  flag = FALSE;
  beg = "000000";
  end = "235959";
  step = 300;
  width = 600;
  // width = { beg = -300; end = 300; }
  grib_code = [ 11, 204, 211 ];
  obs_var   = [];
  type = [ "min", "max", "range", "mean", "stdev", "median", "p80" ];
  vld_freq = 0;
  vld_thresh = 0.0;
}

////////////////////////////////////////////////////////////////////////////////
//
// Settings specific to individual tools
//
////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////
//
// EnsembleStatConfig_default
//
////////////////////////////////////////////////////////////////////////////////

//
// The "ens" entry is a dictionary that specifies the fields for which ensemble
// products should be generated. This is very similar to the "fcst" and "obs"
// entries. This dictionary may include the following entries:
//
//    - The "censor_thresh" and "censor_val" entries are described above.
//
```

```
//    - The "ens_thresh" entry specifies a proportion between 0 and 1 to define
//      the required ratio of valid input ensemble member files. If the ratio
//      of valid input ensemble files to expected ones is too low, the tool
//      will error out.
//
//    - The "vld_thresh" entry specifies a proportion between 0 and 1 to
//      define the required ratio of valid data points. When computing
//      ensemble products, if the ratio of valid data values is too low, the
//      ensemble product will be set to bad data for that point.
//
//    - The "field" entry is as described above. However, in this case, the
//      cat_thresh entry is used for calculating probabilities of exceeding
//      the given threshold. In the default shown below, the probability of
//      accumulated precipitation > 0.0 mm and > 5.0 mm will be calculated
//      from the member accumulated precipitation fields and stored as an
//      ensemble field.
//
ens = {
   censor_thresh = [];
   censor_val    = [];
   ens_thresh    = 1.0;
   vld_thresh    = 1.0;

   field = [
      {
         name       = "APCP";
         level      = "A03";
         cat_thresh = [ >0.0, >=5.0 ];
      }
   ];
}

//
// The nbrhd_prob dictionary defines the neighborhoods used to compute NEP
// and NMEP output. The neighborhood shape is a SQUARE or CIRCLE centered on
// the current point, and the width array specifies the width of the square or
// diameter of the circle as an odd integer. The vld_thresh entry is a number
// between 0 and 1 specifying the required ratio of valid data in the
// neighborhood for an output value to be computed.
//
// If ensemble_flag.nep is set to TRUE, NEP output is created for each
// combination of the categorical threshold (cat_thresh) and neighborhood width
// specified.
```

```
//
nbrhd_prob = {
   width      = [ 5 ];
   shape      = CIRCLE;
   vld_thresh = 0.0;
}

//
// Similar to the interp dictionary, the nmep_smooth dictionary includes a type
// array of dictionaries to define one or more methods for smoothing the NMEP
// data. Setting the interpolation method to nearest neighbor (NEAREST)
// effectively disables this smoothing step.
//
// If ensemble_flag.nmep is set to TRUE, NMEP output is created for each
// combination of the categorical threshold (cat_thresh), neighborhood width
// (nbrhd_prob.width), and smoothing method (nmep_smooth.type) specified.
//
nmep_smooth = {
   vld_thresh      = 0.0;
   shape           = CIRCLE;
   gaussian_dx     = 81.27;
   gaussian_radius = 120;
   type = [
      {
         method = GAUSSIAN;
         width  = 1;
      }
   ];
}

//
// The fcst and obs entries define the fields for which Ensemble-Stat should
// compute rank histograms, probability integral transform histograms,
// spread-skill variance, relative position histograms, economic value, and
// other statistics.
//
// The "ens_ssvar_bin_size" entry sets the width of the variance bins. Smaller
// bin sizes provide the user with more flexibility in how data are binned
// during analysis. The actual variance of the ensemble data will determine the
// number of bins written to the SSVAR output lines.
//
// The "ens_phist_bin_size" is set to a value between 0 and 1. The number of
// bins for the probability integral transform histogram in the PHIST line type
```

```
// is defined as the ceiling of 1.0 / ens_phist_bin_size. For example, a bin
// size of 0.05 results in 20 PHIST bins.
//
// The "prob_cat_thresh" entry is an array of thresholds to be applied in the
// computation of the ranked probability score.  If left empty, but climatology
// data is provided, the climo_cdf thresholds will be used instead.
//
fcst = {
   message_type       = [ "ADPUPA" ];
   ens_ssvar_bin_size = 1;
   ens_phist_bin_size = 0.05;
   prob_cat_thresh    = [];

   field = [
      {
         name  = "APCP";
         level = [ "A03" ];
      }
   ];
}

//
// The "nc_var_str" entry specifies a string for each ensemble field and
// verification task in Ensemble-Stat. This string is parsed from each
// "ens.field" and "obs.field" dictionary entry and is used to customize
// the variable names written to the NetCDF output file. The default is an
// empty string, meaning that no customization is applied to the output variable
// names. When the Ensemble-Stat config file contains two fields with the same
// name and level value, this entry is used to make the resulting variable names
// unique.
// e.g. nc_var_str = "MIN";
//
nc_var_str = "";

//
// The "obs_thresh" entry is an array of thresholds for filtering observation
// values prior to applying ensemble verification logic. The default setting
// of NA means that no observations should be filtered out. Verification output
// will be computed separately for each threshold specified. This option may be
// set separately for each obs.field entry.
//
obs_thresh = [ NA ];
```

```
//
// Setting "skip_const" to true tells Ensemble-Stat to exclude pairs where all
// the ensemble members and the observation have a constant value. For example,
// exclude points with zero precipitation amounts from all output line types.
// This option may be set separately for each obs.field entry. When set to
// false, constant points are included and the observation rank is chosen at
// random.
//
skip_const = FALSE;

//
// Observation error options
// Set dist_type to NONE to use the observation error table instead.
// May be set separately in each "obs.field" entry.
// The obs_error dictionary controls how observation error information should be
// handled. Observation error information can either be specified directly in
// the configuration file or by parsing information from an external table file.
// By default, the MET_BASE/data/table_files/obs_error_table.txt file is read
// but this may be overridden by setting the $MET_OBS_ERROR_TABLE environment
// variable at runtime.
//
// The flag entry toggles the observation error logic on (TRUE) and off (FALSE).
// When flag is TRUE, random observation error perturbations are applied to the
// ensemble member values. No perturbation is applied to the observation values
// but the bias scale and offset values, if specified, are applied.
//
// The dist_type entry may be set to NONE, NORMAL, EXPONENTIAL, CHISQUARED,
// GAMMA, UNIFORM, or BETA. The default value of NONE indicates that the
// observation error table file should be used rather than the configuration
// file settings.
//
// The dist_parm entry is an array of length 1 or 2 specifying the parameters
// for the distribution selected in dist_type. The NORMAL, EXPONENTIAL, and
// CHISQUARED distributions are defined by a single parameter. The GAMMA,
// UNIFORM, and BETA distributions are defined by two parameters. See the GNU
// Scientific Library Reference Manual for more information on these
// distributions:
//    https://www.gnu.org/software/gsl/manual
//
// The inst_bias_scale and inst_bias_offset entries specify bias scale and
// offset values that should be applied to observation values prior to
// perturbing them. These entries enable bias-correction on the fly.
//
```

```
// Defining the observation error information in the configuration file is
// convenient but limited. If defined this way, the random perturbations for all
// points in the current verification task are drawn from the same distribution.
// Specifying an observation error table file instead (by setting dist_type =
// NONE;) provides much finer control, enabling the user to define observation
// error distribution information and bias-correction logic separately for each
// observation variable name, message type, PREPBUFR report type, input report
// type, instrument type, station ID, range of heights, range of pressure
// levels, and range of values.
//
obs_error = {
   flag             = FALSE;   // TRUE or FALSE
   dist_type        = NONE;    // Distribution type
   dist_parm        = [];      // Distribution parameters
   inst_bias_scale  = 1.0;     // Instrument bias scale adjustment
   inst_bias_offset = 0.0;     // Instrument bias offset adjustment
   min              = NA;      // Valid range of data
   max              = NA;
}

//
// The "ensemble_flag" entry is a dictionary of boolean value indicating
// which ensemble products should be generated:
//    - "mean" for the simple ensemble mean
//    - "stdev" for the ensemble standard deviation
//    - "minus" for the mean minus one standard deviation
//    - "plus" for the mean plus one standard deviation
//    - "min" for the ensemble minimum
//    - "max" for the ensemble maximum
//    - "range" for the range of ensemble values
//    - "vld_count" for the number of valid ensemble members
//    - "frequency" for the ensemble relative frequency meeting a threshold
//    - "nep" for the neighborhood ensemble probability
//    - "nmep" for the neighborhood maximum ensemble probability
//    - "rank" to write the rank for the gridded observation field to separate
//      NetCDF output file.
//    - "weight" to write the grid weights specified in grid_weight_flag to the
//      rank NetCDF output file.
//
ensemble_flag = {
   mean      = TRUE;
   stdev     = TRUE;
   minus     = TRUE;
```

```
   plus      = TRUE;
   min       = TRUE;
   max       = TRUE;
   range     = TRUE;
   vld_count = TRUE;
   frequency = TRUE;
   nep       = FALSE;
   nmep      = FALSE;
   rank      = TRUE;
   weight    = FALSE;
}

//
// Random number generator used for random assignment of ranks when they
// are tied.
// http://www.gnu.org/software/gsl/manual/html_node/
// Random-Number-Generator-Performance.html
//
rng = {
   type = "mt19937";
   seed = "";
}

////////////////////////////////////////////////////////////////////////////////
//
// MODEAnalysisConfig_default
//
////////////////////////////////////////////////////////////////////////////////

//
// MODE line options are used to create filters that determine which MODE output
// lines are read in and processed. The MODE line options are numerous. They
// fall into seven categories: toggles, multiple set string options, multiple
// set integer options, integer max/min options, date/time max/min options,
// floating-point max/min options, and miscellaneous options. In order to be
// applied, the options must be uncommented (i.e. remove  the "//" marks) before
// running. These options are described in subsequent sections.
//

//
// Toggles: The MODE line options described in this section are shown in pairs.
// These toggles represent parameters that can have only one (or none) of two
// values. Any of these toggles may be left unspecified. However, if neither
```

```
// option for toggle is indicated, the analysis will produce results that
// combine data from both toggles. This may produce unintended results.
//

//
// This toggle indicates whether forecast or observed lines should be used for
// analysis.
//
fcst      = FALSE;
obs       = FALSE;

//
// This toggle indicates whether single object or object pair lines should be
// used.
//
single    = FALSE;
pair      = FALSE;

//
// This toggle indicates whether simple object or object cluster object lines
// should be used.
//
simple    = FALSE;
cluster   = FALSE;

//
// This toggle indicates whether matched or unmatched object lines should be
// used.
//
matched   = FALSE;
unmatched = FALSE;

//
// Multiple-set string options: The following options set various string
// attributes. They can be set multiple times on the command line but must be
// separated by spaces. Each of these options must be indicated as a string.
// String values that include spaces may be used by enclosing the string in
// quotation marks.
//

//
// This options specifies which model to use
//
```

```
//model    = [];

//
// These two options specify thresholds for forecast and observations objects to
// be used in the analysis, respectively.
//
//fcst_thr = [];
//obs_thr  = [];

//
// These options indicate the names of variables to be used in the analysis for
// forecast and observed fields.
//
//fcst_var = [];
//obs_var = [];

//
// These options indicate vertical levels for forecast and observed fields to be
// used in the analysis.
//
//fcst_lev = [];
//obs_lev = [];

//
// Multiple-set integer options: The following options set various integer
// attributes. Each of the following options may only be indicated as an
// integer.
//

//
// These options are integers of the form HH[MMSS] specifying the lead_time.
//
//fcst_lead       = [];
//obs_lead       = [];

//
// These options are integers of the form HH[MMSS] specifying the valid hour.
//
//fcst_valid_hour = [];
//obs_valid_hour = [];

//
// These options are integers of the form HH[MMSS] specifying the model
```

```
// initialization hour.
//
//fcst_init_hour  = [];
//obs_init_hour  = [];

//
// These options are integers of the form HHMMSS specifying the accumulation
// time.
//
//fcst_accum      = [];
//obs_accum      = [];

//
// These options indicate the convolution radius used for forecast of observed
// objects, respectively.
//
//fcst_rad        = [];
//obs_rad        = [];

//
// Integer max/min options: These options set limits on various integer
// attributes. Leaving a maximum value unset means no upper limit is imposed on
// the value of the attribute. The option works similarly for minimum values.
//

//
// These options are used to indicate minimum/maximum values for the area
// attribute to be used in the analysis.
//
//area_min              = 0;
//area_max              = 0;

//
// These options are used to indicate minimum/maximum values accepted for the
// area thresh. The area thresh is the area of the raw field inside the object
// that meets the threshold criteria.
//
//area_thresh_min       = 0;
//area_thresh_max       = 0;

//
// These options refer to the minimum/maximum values accepted for the
// intersection area attribute.
```

```
//
//intersection_area_min = 0;
//intersection_area_max = 0;

//
// These options refer to the minimum/maximum union area values accepted for
// analysis.
//
//union_area_min        = 0;
//union_area_max        = 0;

//
// These options refer to the minimum/maximum values for symmetric difference
// for objects to be used in the analysis.
//
//symmetric_diff_min    = 0;
//symmetric_diff_max    = 0;

//
// Date/time max/min options: These options set limits on various date/time
// attributes. The values can be specified in one of three ways:  First, the
// options may be indicated by a string of the form YYYMMDD_HHMMSS. This
// specifies a complete calendar date and time. Second, they may be indicated
// by a string of the form YYYYMMMDD_HH. Here, the minutes and seconds are
// assumed to be zero. The third way of indicating date/time attributes is by a
// string of the form YYYMMDD. Here, hours, minutes, and seconds are assumed to
// be zero.
//

//
// These options indicate minimum/maximum values for the forecast valid time.
//
//fcst_valid_min = "";
//fcst_valid_max = "";

//
// These options indicate minimum/maximum values for the observation valid time.
//
//obs_valid_min  = "";
//obs_valid_max  = "";

//
// These options indicate minimum/maximum values for the forecast initialization
```

```
// time.
//
//fcst_init_min  = "";
//fcst_init_max  = "";

//
// These options indicate minimum/maximum values for the observation
// initialization time.
//
//obs_init_min   = "";
//obs_init_max   = "";

//
// Floating-point max/min options: Setting limits on various floating-point
// attributes. One may specify these as integers (i.e., without a decimal
// point), if desired. The following pairs of options indicate minimum and
// maximum values for each MODE attribute that can be described as a floating-
// point number. Please refer to "The MODE Tool" section on attributes in the
// MET User's Guide for a description of these attributes.
//

//centroid_x_min                = 0.0;
//centroid_x_max                = 0.0;

//centroid_y_min                = 0.0;
//centroid_y_max                = 0.0;

//centroid_lat_min              = 0.0;
//centroid_lat_max              = 0.0;

//centroid_lon_min              = 0.0;
//centroid_lon_max              = 0.0;

//axis_ang_min                  = 0.0;
//axis_ang_max                  = 0.0;

//length_min                    = 0.0;
//length_max                    = 0.0;

//width_min                     = 0.0;
//width_max                     = 0.0;

//aspect_ratio_min              = 0.0;
```

```
//aspect_ratio_max              = 0.0;

//curvature_min                 = 0.0;
//curvature_max                 = 0.0;

//curvature_x_min               = 0.0;
//curvature_x_max               = 0.0;

//curvature_y_min               = 0.0;
//curvature_y_max               = 0.0;

//complexity_min                = 0.0;
//complexity_max                = 0.0;

//intensity_10_min              = 0.0;
//intensity_10_max              = 0.0;

//intensity_25_min              = 0.0;
//intensity_25_max              = 0.0;

//intensity_50_min              = 0.0;
//intensity_50_max              = 0.0;

//intensity_75_min              = 0.0;
//intensity_75_max              = 0.0;

//intensity_90_min              = 0.0;
//intensity_90_max              = 0.0;

//intensity_user_min            = 0.0;
//intensity_user_max            = 0.0;

//intensity_sum_min             = 0.0;
//intensity_sum_max             = 0.0;

//centroid_dist_min             = 0.0;
//centroid_dist_max             = 0.0;

//boundary_dist_min             = 0.0;
//boundary_dist_max             = 0.0;

//convex_hull_dist_min          = 0.0;
//convex_hull_dist_max          = 0.0;
```

```
//angle_diff_min                = 0.0;
//angle_diff_max                = 0.0;

//area_ratio_min                = 0.0;
//area_ratio_max                = 0.0;

//intersection_over_area_min    = 0.0;
//intersection_over_area_max    = 0.0;

//complexity_ratio_min          = 0.0;
//complexity_ratio_max          = 0.0;

//percentile_intensity_ratio_min = 0.0;
//percentile_intensity_ratio_max = 0.0;

//interest_min                  = 0.0;
//interest_max                  = 0.0;

////////////////////////////////////////////////////////////////////////////////
//
// MODEConfig_default
//
////////////////////////////////////////////////////////////////////////////////

//
// The "quilt" entry is a boolean to indicate whether all permutations of
// convolution radii and thresholds should be run. If set to false, the number
// of forecast and observation convolution radii and thresholds must all match.
// One configuration of MODE will be run for each group of settings in those
// lists. If set to true, the number of forecast and observation convolution
// radii must match and the number of forecast and observation convolution
// thresholds must match. For N radii and M thresholds, NxM configurations of
// MODE will be run.
//
quilt = false;

//
// The object definition settings for MODE are contained within the "fcst" and
// "obs" entries:
//
//    - The "censor_thresh" and "censor_val" entries are described above.
//      The entries replace the previously supported "raw_thresh" entry.
```

```
//
//    - The "conv_radius" entry specifies the convolution radius in grid
//      squares. The larger the convolution radius, the smoother the objects.
//      Multiple convolution radii may be specified as an array:
//        conv_radius = [ 5, 10, 15 ];
//
//    - The "conv_thresh" entry specifies the convolution threshold used to
//      define MODE objects. The lower the threshold, the larger the objects.
//      Multiple convolution thresholds may be specified as an array:
//         conv_thresh = [ >=5.0, >=10.0, >=15.0 ];
//
//    - The "vld_thresh" entry is described above.
//
//    - The "filter_attr_name" and "filter_attr_thresh" entries are arrays of
//      the same length which specify object filtering criteria. By default, no
//      object filtering criteria is defined.
//
//      The "filter_attr_name" entry is an array of strings specifying the MODE
//      output header column names for the object attributes of interest, such
//      as "AREA", "LENGTH", "WIDTH", and "INTENSITY_50". In addition,
//      "ASPECT_RATIO" specifies the aspect ratio (width/length),
//      "INTENSITY_101" specifies the  mean intensity value, and "INTENSITY_102"
//      specifies the sum of the intensity values.
//
//      The "filter_attr_thresh" entry is an array of thresholds for the
//      object attributes. Any simple objects not meeting all of these
//      filtering criteria are discarded.
//
//      Note that the "area_thresh" and "inten_perc_thresh" entries form
//      earlier versions of MODE are replaced by these options and are now
//      deprecated.
//
//    - The "merge_thresh" entry specifies a lower convolution threshold used
//      when the double-threshold merging method is applied. The number of
//      merge thresholds must match the number of convolution thresholds.
//      Multiple merge thresholds may be specified as an array:
//         merge_thresh = [ >=1.0, >=2.0, >=3.0 ];
//
//    - The "merge_flag" entry specifies the merging methods to be applied:
//       - "NONE" for no merging
//       - "THRESH" for the double-threshold merging method. Merge objects
//         that would be part of the same object at the lower threshold.
//       - "ENGINE" for the fuzzy logic approach comparing the field to itself
```

```
//       - "BOTH" for both the double-threshold and engine merging methods
//
fcst = {
   field = {
      name  = "APCP";
      level = "A03";
   }

   censor_thresh      = [];
   censor_val         = [];
   conv_radius        = 60.0/grid_res; // in grid squares
   conv_thresh        = >=5.0;
   vld_thresh         = 0.5;
   filter_attr_name   = [];
   filter_attr_thresh = [];
   merge_thresh       = >=1.25;
   merge_flag         = THRESH;
}

//
// The "grid_res" entry is the nominal spacing for each grid square in
// kilometers. The variable is not used directly in the code, but subsequent
// variables in the configuration files are defined in terms of it. Therefore,
// setting the appropriately will help ensure that appropriate default values
// are used for these variables.
//
grid_res = 4;

//
// The "match_flag" entry specifies the matching method to be applied:
//    - "NONE" for no matching between forecast and observation objects
//    - "MERGE_BOTH" for matching allowing additional merging in both fields.
//      If two objects in one field match the same object in the other field,
//      those two objects are merged.
//    - "MERGE_FCST" for matching allowing only additional forecast merging
//    - "NO_MERGE" for matching with no additional merging in either field
//
match_flag = MERGE_BOTH;

//
// The "max_centroid_dist" entry specifies the maximum allowable distance in
// grid squares between the centroids of objects for them to be compared.
// Setting this to a reasonable value speeds up the runtime enabling MODE to
```

```
// skip unreasonable object comparisons.
//
max_centroid_dist = 800.0/grid_res;

//
// The weight variables control how much weight is assigned to each pairwise
// attribute when computing a total interest value for object pairs. The weights
// need not sum to any particular value but must be non-negative. When the
// total interest value is computed, the weighted sum is normalized by the
// sum of the weights listed.
//
weight = {
   centroid_dist    = 2.0;
   boundary_dist    = 4.0;
   convex_hull_dist = 0.0;
   angle_diff       = 1.0;
   area_ratio       = 1.0;
   int_area_ratio   = 2.0;
   complexity_ratio = 0.0;
   inten_perc_ratio = 0.0;
   inten_perc_value = 50;
}

//
// The set of interest function variables listed define which values are of
// interest for each pairwise attribute measured. The interest functions may be
// defined as a piecewise linear function or as an algebraic expression. A
// piecewise linear function is defined by specifying the corner points of its
// graph. An algebraic function may be defined in terms of several built-in
// mathematical functions.
//
interest_function = {

   centroid_dist = (
      (            0.0, 1.0 )
      (  60.0/grid_res, 1.0 )
      ( 600.0/grid_res, 0.0 )
   );

   boundary_dist = (
      (            0.0, 1.0 )
      ( 400.0/grid_res, 0.0 )
   );
```

```
   convex_hull_dist = (
      (            0.0, 1.0 )
      ( 400.0/grid_res, 0.0 )
   );

   angle_diff = (
      (  0.0, 1.0 )
      ( 30.0, 1.0 )
      ( 90.0, 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 )
   );

   complexity_ratio = ratio_if;

   inten_perc_ratio = ratio_if;
}

//
// The total_interest_thresh variable should be set between 0 and 1. This
// threshold is applied to the total interest values computed for each pair of
// objects and is used in determining matches.
//
total_interest_thresh = 0.7;

//
// The print_interest_thresh variable determines which pairs of object
// attributes will be written to the output object attribute ASCII file. The
```

```
// user may choose to set the print_interest_thresh to the same value as the
// total_interest_thresh, meaning that only object pairs that actually match are
// written to the output file. When set to zero, all object pair attributes will
// be written as long as the distance between the object centroids is less than
// the max_centroid_dist variable.
//
print_interest_thresh = 0.0;

//
// When applied, the plot_valid_flag variable indicates that only the region
// containing valid data after masking is applied should be plotted. TRUE
// indicates the entire domain should be plotted; FALSE indicates only the
// region containing valid data after masking should be plotted.
//
plot_valid_flag = FALSE;

//
// When applied, the plot_gcarc_flag variable indicates that the edges of
// polylines should be plotted using great circle arcs as opposed to straight
// lines in the grid.
//
plot_gcarc_flag = FALSE;

//
// The ct_stats_flag can be set to TRUE or FALSE to produce additional output,
// in the form of contingency table counts and statistics.
//
ct_stats_flag = TRUE;

//
// When MODE is run on global grids, this parameter specifies how many grid
// squares to shift the grid to the right. MODE does not currently connect
// objects from one side of a global grid to the other, potentially causing
// objects straddling that longitude to be cut in half. Shifting the grid by
// some amount enables the user to control where that longitude cut line occurs.
// This option provides a very specialized case of automated regridding. The
// much more flexible "regrid" option may be used instead.
//
shift_right = 0;

////////////////////////////////////////////////////////////////////////////////
//
// PB2NCConfig_default
```

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

//
// The PB2NC tool filters out observations from PREPBUFR or BUFR files using the
// following criteria:
//  (1) by message type: supply a list of PREPBUFR message types to retain
//  (2) by station id: supply a list of observation stations to retain
//  (3) by valid time: supply the beginning and ending time offset values
//      in the obs_window entry described above.
//  (4) by location: use the "mask" entry described below to supply either
//      an NCEP masking grid, a masking lat/lon polygon or a file to a
//      mask lat/lon polygon
//  (5) by elevation: supply min/max elevation values
//  (6) by report type: supply a list of report types to retain using
//      pb_report_type and in_report_type entries described below
//  (7) by instrument type: supply a list of instrument type to
//      retain
//  (8) by vertical level: supply beg/end vertical levels using the
//      level_range entry described below
//  (9) by variable type: supply a list of observation variable types to
//      retain using the obs_bufr_var entry described below
// (11) by quality mark: supply a quality mark threshold
// (12) Flag to retain values for all quality marks, or just the first
//      quality mark (highest): use the event_stack_flag described below
// (13) by data level category: supply a list of category types to
//      retain.
//
//      0 - Surface level (mass reports only)
//      1 - Mandatory level (upper-air profile reports)
//      2 - Significant temperature level (upper-air profile reports)
//      2 - Significant temperature and winds-by-pressure level
//          (future combined mass and wind upper-air reports)
//      3 - Winds-by-pressure level (upper-air profile reports)
//      4 - Winds-by-height level (upper-air profile reports)
//      5 - Tropopause level (upper-air profile reports)
//      6 - Reports on a single level
//          (e.g., aircraft, satellite-wind, surface wind,
//           precipitable water retrievals, etc.)
//      7 - Auxiliary levels generated via interpolation from spanning levels
//          (upper-air profile reports)
//
```

```
//
// In the PB2NC tool, the "message_type" entry is an array of message types
// to be retained. An empty list indicates that all should be retained.
//
// List of valid message types:
//    ADPUPA AIRCAR AIRCFT ADPSFC ERS1DA GOESND GPSIPW
//    MSONET PROFLR QKSWND RASSDA SATEMP SATWND SFCBOG
//    SFCSHP SPSSMI SYNDAT VADWND
//
//    e.g. message_type[] = [ "ADPUPA", "AIRCAR" ];
//
// http://www.emc.ncep.noaa.gov/mmb/data_processing/prepbufr.doc/table_1.htm
//
message_type = [];

//
// Mapping of message type group name to comma-separated list of values.
// The default setting defines ANYAIR, ANYSFC, and ONLYSF as groups.
// 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";                     }
];

//
// The "station_id" entry is an array of station ids to be retained or
// the filename which contains station ids. An array of station ids
// contains a comma-separated list. An empty list indicates that all
// stations should be retained.
//
// e.g. station_id = [ "KDEN" ];
//
station_id = [];

//
// The "elevation_range" entry is a dictionary which contains "beg" and "end"
// entries specifying the range of observing locations elevations to be
// retained.
//
elevation_range = {
```

```
   beg =  -1000;
   end = 100000;
}

//
// The "pb_report_type" entry is an array of PREPBUFR report types to be
// retained. The numeric "pb_report_type" entry allows for further
// stratification within message types. An empty list indicates that all should
// be retained.
//
// http://www.emc.ncep.noaa.gov/mmb/data_processing/prepbufr.doc/table_4.htm
//
// e.g.
//   Report Type 120 is for message type ADPUPA but is only RAWINSONDE
//   Report Type 132 is for message type ADPUPA but is only FLIGHT-LEVEL RECON
//     and PROFILE DROPSONDE
//
pb_report_type  = [];

//
// The "in_report_type" entry is an array of input report type values to be
// retained. The numeric "in_report_type" entry provides additional
// stratification of observations. An empty list indicates that all should
// be retained.
//
// http://www.emc.ncep.noaa.gov/mmb/data_processing/prepbufr.doc/table_6.htm
//
// e.g.
// Input Report Type 11 Fixed land RAOB and PIBAL by block and station number
// Input Report Type 12 Fixed land RAOB and PIBAL by call letters

in_report_type = [];

//
// The "instrument_type" entry is an array of instrument types to be retained.
// An empty list indicates that all should be retained.
//
instrument_type = [];

//
// The "level_range" entry is a dictionary which contains "beg" and "end"
// entries specifying the range of vertical levels (1 to 255) to be retained.
//
```

```
level_range = {
   beg = 1;
   end = 255;
}

//
// The "level_category" entry is an array of integers specifying which level
// categories should be retained:
//    0 = Surface level (mass reports only)
//    1 = Mandatory level (upper-air profile reports)
//    2 = Significant temperature level (upper-air profile reports)
//    2 = Significant temperature and winds-by-pressure level
//        (future combined mass and wind upper-air reports)
//    3 = Winds-by-pressure level (upper-air profile reports)
//    4 = Winds-by-height level (upper-air profile reports)
//    5 = Tropopause level (upper-air profile reports)
//    6 = Reports on a single level
//        (e.g., aircraft, satellite-wind, surface wind,
//         precipitable water retrievals, etc.)
//    7 = Auxiliary levels generated via interpolation from spanning levels
//        (upper-air profile reports)
// An empty list indicates that all should be retained.
//
// http://www.emc.ncep.noaa.gov/mmb/data_processing/prepbufr.doc/table_1.htm
//
level_category = [];

//
// The "obs_bufr_var" entry is an array of strings containing BUFR variable
// names to be retained or derived. This replaces the "obs_grib_code" setting
// from earlier versions of MET. Run PB2NC on your data with the "-index"
// command line option to see the list of available observation variables.
//
// Observation variables that can be derived begin with "D_":
//    D_DPT   for Dew point Temperature in K
//    D_WDIR  for Wind Direction
//    D_WIND  for Wind Speed in m/s
//    D_RH    for Relative Humidity in %
//    D_MIXR  for Humidity Mixing Ratio in kg/kg
//    D_PRMSL for Pressure Reduced to Mean Sea Level in Pa
//
obs_bufr_var = [ "QOB", "TOB", "ZOB", "UOB", "VOB" ];
```

```
//
// Mapping of input BUFR variable names to output variables names.
// The default PREPBUFR map, obs_prepbufr_map, is appended to this map.
// Users may choose to rename BUFR variables to match the naming convention
// of the forecast the observation is used to verify.
//
obs_bufr_map = [];

//
// Default mapping for PREPBUFR. Replace input BUFR variable names with GRIB
// abbreviations in the output. This default map is appended to obs_bufr_map.
// This should not typically be overridden. This default mapping provides
// backward-compatibility for earlier versions of MET which wrote GRIB
// abbreviations to the output.
//
obs_prefbufr_map = [
   { key = "POB";     val = "PRES";  },
   { key = "QOB";     val = "SPFH";  },
   { key = "TOB";     val = "TMP";   },
   { key = "ZOB";     val = "HGT";   },
   { key = "UOB";     val = "UGRD";  },
   { key = "VOB";     val = "VGRD";  },
   { key = "D_DPT";   val = "DPT";   },
   { key = "D_WDIR";  val = "WDIR";  },
   { key = "D_WIND";  val = "WIND";  },
   { key = "D_RH";    val = "RH";    },
   { key = "D_MIXR";  val = "MIXR";  },
   { key = "D_PRMSL"; val = "PRMSL"; }
];

//
// The "quality_mark_thresh" entry specifies the maximum quality mark value
// to be retained. Observations with a quality mark LESS THAN OR EQUAL TO
// this threshold will be retained, while observations with a quality mark
// GREATER THAN this threshold will be discarded.
//
// http://www.emc.ncep.noaa.gov/mmb/data_processing/prepbufr.doc/table_7.htm
//
quality_mark_thresh = 2;

//
// The "event_stack_flag" entry is set to "TOP" or "BOTTOM" to
// specify whether observations should be drawn from the top of the event
```

```
// stack (most quality controlled) or the bottom of the event stack (most raw).
//
event_stack_flag = TOP;

////////////////////////////////////////////////////////////////////////////////
//
// SeriesAnalysisConfig_default
//
////////////////////////////////////////////////////////////////////////////////

//
// Computation may be memory intensive, especially for large grids.
// The "block_size" entry sets the number of grid points to be processed
// concurrently (i.e. in one pass through a time series). Smaller values
// require less memory but increase the number of passes through the data.
//
block_size = 1024;

//
// Ratio of valid matched pairs to total length of series for a grid
// point. If valid threshold is exceeded at that grid point the statistics
// are computed and stored. If not, a bad data flag is stored. The default
// setting requires all data in the series to be valid.
//
//
vld_thresh = 1.0;

//
// Statistical output types need to be specified explicitly. Refer to User's
// Guide for available output types. To keep output file size reasonable,
// it is recommended to process a few output types at a time, especially if the
// grid is large.
//
output_stats = {
   fho    = [];
   ctc    = [];
   cts    = [];
   mctc   = [];
   mcts   = [];
   cnt    = [ "RMSE", "FBAR", "OBAR" ];
   sl1l2  = [];
   pct    = [];
   pstd   = [];
```

```
   pjc    = [];
   prc    = [];
}

////////////////////////////////////////////////////////////////////////////////
//
// STATAnalysisConfig_default
//
////////////////////////////////////////////////////////////////////////////////

//
// The "jobs" entry is an array of STAT-Analysis jobs to be performed.
// Each element in the array contains the specifications for a single analysis
// job to be performed. The format for an analysis job is as follows:
//
//    -job job_name
//    OPTIONAL ARGS
//
//    Where "job_name" is set to one of the following:
//
//       "filter"
//          To filter out the STAT or TCMPR lines matching the job filtering
//          criteria specified below and using the optional arguments below.
//          The output STAT lines are written to the file specified using the
//          "-dump_row" argument.
//          Required Args: -dump_row
//
//       "summary"
//          To compute summary information for a set of statistics.
//          The summary output includes the mean, standard deviation,
//          percentiles (0th, 10th, 25th, 50th, 75th, 90th, and 100th), range,
//          and inter-quartile range. Also included are columns summarizing the
//          computation of WMO mean values. Both unweighted and weighted mean
//          values are reported, and they are computed using three types of
//          logic:
//             - simple arithmetic mean (default)
//             - square root of the mean of the statistic squared
//               (applied to columns listed in "wmo_sqrt_stats")
//             - apply fisher transform
//                (applied to columns listed in "wmo_fisher_stats")
//          The columns of data to be summarized are specified in one of two
//          ways:
//             - Specify the -line_type option once and specify one or more
```

```
//               -column names.
//            - Format the -column option as LINE_TYPE:COLUMN.
//
//          Use the -derive job command option to automatically derive
//          statistics on the fly from input contingency tables and partial
//          sums.
//
//          Use the -column_union TRUE/FALSE job command option to compute
//          summary statistics across the union of input columns rather than
//          processing them separately.
//
//          For TCStat, the "-column" argument may be set to:
//             "TRACK" for track, along-track, and cross-track errors.
//             "WIND" for all wind radius errors.
//             "TI" for track and maximum wind intensity errors.
//             "AC" for along-track and cross-track errors.
//             "XY" for x-track and y-track errors.
//             "col" for a specific column name.
//             "col1-col2" for a difference of two columns.
//             "ABS(col or col1-col2)" for the absolute value.
//
//          Required Args: -line_type, -column
//          Optional Args: -by column_name to specify case information
//                         -out_alpha to override default alpha value of 0.05
//                         -derive to derive statistics on the fly
//                         -column_union to summarize multiple columns
//
//       "aggregate"
//          To aggregate the STAT data for the STAT line type specified using
//          the "-line_type" argument. The output of the job will be in the
//          same format as the input line type specified. The following line
//          types may be aggregated:
//          -line_type FHO, CTC, MCTC,
//                     SL1L2, SAL1L2, VL1L2, VAL1L2,
//                     PCT, NBRCNT, NBRCTC, GRAD,
//                     ISC, ECNT, RPS, RHIST, PHIST, RELP, SSVAR
//          Required Args: -line_type
//
//       "aggregate_stat"
//          To aggregate the STAT data for the STAT line type specified using
//          the "-line_type" argument. The output of the job will be the line
//          type specified using the "-out_line_type" argument. The valid
//          combinations of "-line_type" and "-out_line_type" are listed below.
```

```
//          -line_type FHO,   CTC,    -out_line_type CTS, ECLV
//          -line_type MCTC           -out_line_type MCTS
//          -line_type SL1L2, SAL1L2, -out_line_type CNT
//          -line_type VL1L2          -out_line_type VCNT
//          -line_type VL1L2, VAL1L2, -out_line_type WDIR (wind direction)
//          -line_type PCT,           -out_line_type PSTD, PJC, PRC, ECLV
//          -line_type NBRCTC,        -out_line_type NBRCTS
//          -line_type ORANK,         -out_line_type ECNT, RPS, RHIST, PHIST,
//                                                   RELP, SSVAR
//          -line_type MPR,           -out_line_type FHO, CTC, CTS,
//                                                   MCTC, MCTS, CNT,
//                                                   SL1L2, SAL1L2,
//                                                   VL1L2, VCNT,
//                                                   PCT, PSTD, PJC, PRC, ECLV,
//                                                   WDIR (wind direction)
//          Required Args:
//             -line_type, -out_line_type
//          Additional Required Args for -line_type MPR:
//             -out_thresh or -out_fcst_thresh and -out_obs_thresh
//             When -out_line_type FHO, CTC, CTS, MCTC, MCTS,
//                                  PCT, PSTD, PJC, PRC
//          Additional Optional Args for -line_type MPR:
//             -mask_grid, -mask_poly, -mask_sid
//             -out_thresh or -out_fcst_thresh and -out_obs_thresh
//             -out_cnt_logic
//             -out_wind_thresh or -out_fcst_wind_thresh and
//             -out_obs_wind_thresh
//             -out_wind_logic
//             When -out_line_type WDIR
//          Additional Optional Arg for:
//             -line_type ORANK -out_line_type PHIST, SSVAR ...
//             -out_bin_size
//          Additional Optional Args for:
//             -out_line_type ECLV ...
//             -out_eclv_points
//
//       "ss_index"
//          The skill score index job can be configured to compute a weighted
//          average of skill scores derived from a configurable set of
//          variables, levels, lead times, and statistics. The skill score
//          index is computed using two models, a forecast model and a
//          reference model. For each statistic in the index, a skill score
//          is computed as:
```

```
//             SS = 1 - (S[model]*S[model])/(S[reference]*S[reference])
//          Where S is the statistic.
//          Next, a weighted average is computed over all the skill scores.
//          Lastly, an index value is computed as:
//             Index = sqrt(1/(1-SS[avg]))
//          Where SS[avg] is the weighted average of skill scores.
//          Required Args:
//             Exactly 2 entries for -model, the forecast model and reference
//             For each term of the index:
//             -fcst_var, -fcst_lev, -fcst_lead, -line_type, -column, -weight
//             Where -line_type is CNT or CTS and -column is the statistic.
//             Optionally, specify other filters for each term, -fcst_thresh.
//
//       "go_index"
//          The GO Index is a special case of the skill score index consisting
//          of a predefined set of variables, levels, lead times, statistics,
//          and weights.
//          For lead times of 12, 24, 36, and 48 hours, it contains RMSE for:
//          - Wind Speed at the surface(b), 850(a), 400(a), 250(a) mb
//          - Dew point Temperature at the surface(b), 850(b), 700(b), 400(b) mB
//          - Temperature at the surface(b), 400(a) mB
//          - Height at 400(a) mB
//          - Sea Level Pressure(b)
//          Where (a) means weights of 4, 3, 2, 1 for the lead times, and
//          (b) means weights of 8, 6, 4, 2 for the lead times.
//
//          Required Args: None
//
//       "ramp"
//          The ramp job operates on a time-series of forecast and observed
//          values and is analogous to the RIRW (Rapid Intensification and
//          Weakening) job supported by the tc_stat tool. The amount of change
//          from one time to the next is computed for forecast and observed
//          values. Those changes are thresholded to define events which are
//          used to populate a 2x2 contingency table.
//
//          Required Args:
//             -ramp_thresh (-ramp_thresh_fcst or -ramp_thresh_obs)
//                For DYDT, threshold for the amount of change required to
//                define an event.
//                For SWING, threshold the slope.
//             -swing_width val
//                Required for the swinging door algorithm width.
```

```
//
//          Optional Args:
//             -ramp_type str
//                Overrides the default ramp definition algorithm to be used.
//                May be set to DYDT (default) or SWING for the swinging door
//                algorithm.
//             -line_type str
//                Overrides the default input line type, MPR.
//             -out_line_type str
//                Overrides the default output line types of CTC and CTS.
//                Set to CTC,CTS,MPR for all possible output types.
//             -column fcst_column,obs_column
//                Overrides the default forecast and observation columns
//                to be used, FCST and OBS.
//             -ramp_time HH[MMSS] (-ramp_time_fcst or -ramp_time_obs)
//                Overrides the default ramp time interval, 1 hour.
//             -ramp_exact true/false (-ramp_exact_fcst or -ramp_exact_obs)
//                Defines ramps using an exact change (true, default) or maximum
//                change in the time window (false).
//             -ramp_window width in HH[MMSS] format
//             -ramp_window beg end in HH[MMSS] format
//                Defines a search time window when attempting to convert misses
//                to hits and false alarms to correct negatives. Use 1 argument
//                to define a symmetric time window or 2 for an asymmetric
//                window. Default window is 0 0, requiring an exact match.
//
//    Job command FILTERING options to further refine the STAT data:
//       Each optional argument may be used in the job specification multiple
//       times unless otherwise indicated. When multiple optional arguments of
//       the same type are indicated, the analysis will be performed over their
//       union:
//
//       "-model            name"
//       "-fcst_lead        HHMMSS"
//       "-obs_lead         HHMMSS"
//       "-fcst_valid_beg   YYYYMMDD[_HH[MMSS]]" (use once)
//       "-fcst_valid_end   YYYYMMDD[_HH[MMSS]]" (use once)
//       "-obs_valid_beg    YYYYMMDD[_HH[MMSS]]" (use once)
//       "-obs_valid_end    YYYYMMDD[_HH[MMSS]]" (use once)
//       "-fcst_init_beg    YYYYMMDD[_HH[MMSS]]" (use once)
//       "-fcst_init_end    YYYYMMDD[_HH[MMSS]]" (use once)
//       "-obs_init_beg     YYYYMMDD[_HH[MMSS]]" (use once)
//       "-obs_init_end     YYYYMMDD[_HH[MMSS]]" (use once)
```

```
//       "-fcst_init_hour   HH[MMSS]"
//       "-obs_init_hour    HH[MMSS]"
//       "-fcst_valid_hour" HH[MMSS]
//       "-obs_valid_hour"  HH[MMSS]
//       "-fcst_var         name"
//       "-obs_var          name"
//       "-fcst_lev         name"
//       "-obs_lev          name"
//       "-obtype           name"
//       "-vx_mask          name"
//       "-interp_mthd      name"
//       "-interp_pnts      n"
//       "-fcst_thresh      t"
//       "-obs_thresh       t"
//       "-cov_thresh       t"
//       "-thresh_logic     UNION, or, ||
//                          INTERSECTION, and, &&
//                          SYMDIFF, symdiff, *
//       "-alpha            a"
//       "-line_type        type"
//       "-column           name"
//       "-weight           value"
//
//   Job command FILTERING options that may be used only when -line_type
//   has been listed once. These options take two arguments: the name of the
//   data column to be used and the min, max, or exact value for that column.
//   If multiple column eq/min/max/str options are listed, the job will be
//   performed on their intersection:
//
//       "-column_min    col_name value"     e.g. -column_min BASER 0.02
//       "-column_max    col_name value"
//       "-column_eq     col_name value"
//       "-column_thresh col_name threshold" e.g. -column_thresh FCST '>273'
//       "-column_str    col_name string" separate multiple filtering strings
//                                        with commas
//
//   Job command options to DEFINE the analysis job. Unless otherwise noted,
//   these options may only be used ONCE per analysis job:
//
//       "-dump_row        path"
//
//       "-mask_grid       name"
//       "-mask_poly       file"
```

```
//       "-mask_sid        file|list" see description of "sid" entry above
//
//       "-out_line_type   name"
//       "-out_thresh      value" sets both -out_fcst_thresh and -out_obs_thresh
//       "-out_fcst_thresh value" multiple for multi-category contingency tables
//                                and probabilistic forecasts
//       "-out_obs_thresh  value" multiple for multi-category contingency tables
//       "-out_cnt_logic   value"
//
//       "-out_wind_thresh      value"
//       "-out_fcst_wind_thresh value"
//       "-out_obs_wind_thresh  value"
//       "-out_wind_logic       value"
//
//       "-out_bin_size    value"
//
//       "-out_eclv_points value" see description of "eclv_points" config file
//                                entry
//
//       "-out_alpha       value"
//
//       "-boot_interval   value"
//       "-boot_rep_prop   value"
//       "-n_boot_rep      value"
//       "-boot_rng        value"
//       "-boot_seed       value"
//
//       "-rank_corr_flag  value"
//       "-vif_flag        value"
//
//   For aggregate and aggregate_stat job types:
//
//       "-out_stat        path"   to write a .stat output file for the job
//                                 including the .stat header columns. Multiple
//                                 values for each header column are written as
//                                 a comma-separated list.
//       "-set_hdr col_name value" may be used multiple times to explicity
//                                 specify what should be written to the header
//                                 columns of the output .stat file.
//
//   When using the "-by" job command option, you may reference those columns
//   in the "-set_hdr" job command options. For example, when computing statistics
//   separately for each station, write the station ID string to the VX_MASK column
```

```
//   of the output .stat output file:
//     -job aggregate_stat -line_type MPR -out_line_type CNT \
//     -by OBS_SID -set_hdr VX_MASK OBS_SID -stat_out out.stat
//   When using mulitple "-by" options, use "CASE" to reference the full string:
//     -by FCST_VAR,OBS_SID -set_hdr DESC CASE -stat_out out.stat
//
jobs = [
   "-job filter         -line_type SL1L2 -vx_mask DTC165 \
    -dump_row  job_filter_SL1L2.stat",
   "-job summary        -line_type CNT   -alpha 0.050 -fcst_var TMP \
    -dump_row job_summary_ME.stat -column ME",
   "-job aggregate      -line_type SL1L2 -vx_mask DTC165 -vx_mask DTC166 \
    -fcst_var TMP -dump_row job_aggregate_SL1L2_dump.stat  \
    -out_stat job_aggregate_SL1L2_out.stat \
    -set_hdr VX_MASK CONUS",
   "-job aggregate_stat -line_type SL1L2 -out_line_type CNT -vx_mask DTC165  \
    -vx_mask DTC166 -fcst_var TMP \
    -dump_row  job_aggregate_stat_SL1L2_CNT_in.stat",
   "-job aggregate_stat -line_type MPR   -out_line_type CNT -vx_mask DTC165 \
    -vx_mask DTC166 -fcat_var TMP -dump_row job_aggregate_stat_MPR_CNT_in.stat",
   "-job aggregate      -line_type CTC   -fcst_thresh <300.000 -vx_mask DTC165 \
    -vx_mask DTC166 -fcst_var TMP -dump_row job_aggregate_CTC_in.stat",
   "-job aggregate_stat -line_type CTC   -out_line_type CTS \
    -fcst_thresh <300.000 -vx_mask DTC165 -vx_mask DTC166 -fcst_var TMP \
    -dump_row job_aggregate_stat_CTC_CTS_in.stat",
   "-job aggregate      -line_type MCTC  -column_eq N_CAT 4 -vx_mask DTC165 \
    -vx_mask DTC166 -fcst_var APCP_24 -dump_row job_aggregate_MCTC_in.stat",
   "-job aggregate_stat -line_type MCTC  -out_line_type MCTS \
    -column_eq N_CAT 4 -vx_mask DTC165 -vx_mask DTC166 -fcst_var APCP_24 \
    -dump_row job_aggregate_stat_MCTC_MCTS_in.stat",
   "-job aggregate      -line_type PCT   -vx_mask DTC165 -vx_mask DTC166 \
    -dump_row job_aggregate_PCT_in.stat",
   "-job aggregate_stat -line_type PCT   -out_line_type PSTD -vx_mask DTC165 \
    -vx_mask DTC166 -dump_row job_aggregate_stat_PCT_PSTD_in.stat",
   "-job aggregate      -line_type ISC   -fcst_thresh >0.000 -vx_mask TILE_TOT \
    -fcst_var APCP_12 -dump_row job_aggregate_ISC_in.stat",
   "-job aggregate      -line_type RHIST -obtype MC_PCP -vx_mask HUC4_1605 \
    -vx_mask HUC4_1803 -dump_row job_aggregate_RHIST_in.stat",
   "-job aggregate      -line_type SSVAR -obtype MC_PCP -vx_mask HUC4_1605 \
    -vx_mask HUC4_1803 -dump_row job_aggregate_SSVAR_in.stat",
   "-job aggregate_stat -line_type ORANK -out_line_type RHIST -obtype ADPSFC \
    -vx_mask HUC4_1605 -vx_mask HUC4_1803 \
    -dump_row job_aggregate_stat_ORANK_RHIST_in.stat"
```

```
];

//
// List of statistics by the logic that should be applied when computing their
// WMO mean value in the summary job. Each entry is a line type followed by the
// statistic name. Statistics using the default arithemtic mean method do not
// need to be listed.
//
wmo_sqrt_stats   = [];
wmo_fisher_stats = [];

//
// The "vif_flag" entry is a boolean to indicate whether a variance inflation
// factor should be computed when aggregating a time series of contingency
// table counts or partial sums. The VIF is used to adjust the normal
// confidence intervals computed for the aggregated statistics.
//
vif_flag = FALSE;

////////////////////////////////////////////////////////////////////////////////
//
// WaveletStatConfig_default
//
////////////////////////////////////////////////////////////////////////////////

//
// The "grid_decomp_flag" entry specifies how the grid should be decomposed in
// Wavelet-Stat into dyadic (2^n x 2^n) tiles:
//    - "AUTO" to tile the input data using tiles of dimension n by n where n
//      is the largest integer power of 2 less than the smallest dimension of
//      the input data. Center as many tiles as possible with no overlap.
//    - "TILE" to use the tile definition specified below.
//    - "PAD" to pad the input data out to the nearest integer power of 2.
//
grid_decomp_flag = AUTO;

//
// The "tile" entry is a dictionary that specifies how tiles should be defined
// in Wavelet-Stat when the "grid_decomp_flag" is set to "TILE":
//
//    - The "width" entry specifies the dimension for all tiles and must be
//      an integer power of 2.
//
```

```
//    - The "location" entry is an array of dictionaries where each element
//      consists of an "x_ll" and "y_ll" entry specifying the lower-left (x,y)
//      coordinates of the tile.
//
tile = {
   width = 0;
   location = [
      {
         x_ll = 0;
         y_ll = 0;
      }
   ];
}

//
// The "wavelet" entry is a dictionary in Wavelet-Stat that specifies how the
// wavelet decomposition should be performed:
//
//    - The "type" entry specifies which wavelet should be used.
//
//    - The "member" entry specifies the wavelet shape.
//      http://www.gnu.org/software/gsl/manual/html_node/DWT-Initialization.html
//
//    - Valid combinations of the two are listed below:
//       - "HAAR" for Haar wavelet (member = 2)
//       - "HAAR_CNTR" for Centered-Haar wavelet (member = 2)
//       - "DAUB" for Daubechies wavelet (member = 4, 6, 8, 10, 12, 14, 16,
//         18, 20)
//       - "DAUB_CNTR" for Centered-Daubechies wavelet (member = 4, 6, 8, 10,
//          12, 14, 16, 18, 20)
//       - "BSPLINE" for Bspline wavelet (member = 103, 105, 202, 204, 206,
//         208, 301, 303, 305, 307, 309)
//       - "BSPLINE_CNTR" for Centered-Bspline wavelet (member = 103, 105, 202,
//          204, 206, 208, 301, 303, 305, 307, 309)
//
wavelet = {
   type   = HAAR;
   member = 2;
}

//
// The "obs_raw_plot", "wvlt_plot", and "object_plot" entries are dictionaries
// similar to the "fcst_raw_plot" described in the "Settings common to multiple
```

```
// tools" section.
//

////////////////////////////////////////////////////////////////////////
//
// WWMCARegridConfig_default
//
////////////////////////////////////////////////////////////////////////

//
// Specify the grid to which the data should be interpolated in one of the
// following ways:
//
//    - Name ("GNNN" where NNN indicates the three digit NCEP grid number)
//
//    - lambert Nx Ny lat_ll lon_ll lon_orient D_km R_km standard_parallel_1
//      [standard_parallel_2] N|S
//
//    - stereo Nx Ny lat_ll lon_ll lon_orient D_km R_km lat_scale N|S
//
//    - latlon Nx Ny lat_ll lon_ll delta_lat delta_lon
//
//    - mercator Nx Ny lat_ll lon_ll lat_ur lon_ur
//
//    - gaussian lon_zero Nx Ny
//
to_grid = "lambert 614 428 12.190 -133.459 -95.0 12.19058 6367.47 25.0 N";

//
// Supply the NetCDF output information
//
// e.g. variable_name = "Cloud_Pct";
//      units         = "percent";
//      long_name     = "cloud cover percent";
//      level         = "SFC";
//
variable_name = "";
units         = "";
long_name     = "";
level         = "";

//
// Maximum pixel age in minutes
```

```
//
max_minutes = 120;

//
// The WWMCA pixel age data is stored in binary data files in 4-byte blocks.
// The swap_endian option indicates whether the endian-ness of the data should
// be swapped after reading.
//
swap_endian = TRUE;

//
// By default, wwmca_regrid writes the cloud percent data specified on the
// command line to the output file. This option writes the pixel age data,
// in minutes, to the output file instead.
//
write_pixel_age = FALSE;
```

3.5.2 MET-TC Configuration File Options

The information listed below may also be found in the data/config/README_TC file.

```
////////////////////////////////////////////////////////////////////////////////
//
// Configuration file overview.
//
////////////////////////////////////////////////////////////////////////////////

See README for configuration file overview.

////////////////////////////////////////////////////////////////////////////////
//
// Configuration settings common to multiple tools
//
////////////////////////////////////////////////////////////////////////////////

//
// Specify a comma-separated list of storm id's to be used:
//    2-letter basin, 2-digit cyclone number, 4-digit year
// An empty list indicates that all should be used.
//
// e.g. storm_id = [ "AL092011" ];
//
```

```
// This may also be set using basin, cyclone, and timing information below.
//
storm_id = [];

//
// Specify a comma-separated list of basins to be used.
// Expected format is 2-letter basin identifier.
// An empty list indicates that all should be used.
//   Valid basins: WP, IO, SH, CP, EP, AL, SL
//
// e.g. basin = [ "AL", "EP" ];
//
basin = [];

//
// Specify a comma-separated list of cyclone numbers (01-99) to be used.
// An empty list indicates that all should be used.
//
// e.g. cyclone = [ "01", "02", "03" ];
//
cyclone = [];


//
// Specify a comma-separated list of storm names to be used.
// An empty list indicates that all should be used.
//
// e.g. storm_name = [ "KATRINA" ];
//
storm_name = [];

//
// Specify a model initialization time window in YYYYMMDD[_HH[MMSS]] format
// or provide a list of specific initialization times to include (inc)
// or exclude (exc). Tracks whose initial time meets the specified
// criteria will be used. An empty string indicates that all times
// should be used.
//
// e.g. init_beg = "20100101";
//      init_end = "20101231";
//      init_inc = [ "20101231_06" ];
//      init_exc = [ "20101231_00" ];
//
```

```
init_beg = "";
init_end = "";
init_inc = [];
init_exc = [];

//
// Specify a model valid time window in YYYYMMDD[_HH[MMSS]] format.
// Tracks for which all valid times fall within the time window will be used.
// An empty string indicates that all times should be used.
//
// e.g. valid_beg = "20100101";
//      valid_end = "20101231";
//
valid_beg = "";
valid_end = "";

//
// Specify a comma-separated list of model initialization hours to be used
// in HH[MMSS] format. An empty list indicates that all hours should be used.
//
// e.g. init_hour = [ "00", "06", "12", "18" ];
//
init_hour = [];

//
// Specify the required lead time in HH[MMSS] format.
// Tracks that contain all of these required times will be
// used. If a track has additional lead times, it will be
// retained.  An empty list indicates that no lead times
// are required to determine which tracks are to be used;
// all lead times will be used.
//
lead_req  = [];

//
// Specify lat/lon polylines defining masking regions to be applied.
// Tracks whose initial location falls within init_mask will be used.
// Tracks for which all locations fall within valid_mask will be used.
//
// e.g. init_mask  = "MET_BASE/poly/EAST.poly";
//
init_mask  = "";
valid_mask = "";
```

```
//
// Indicate the version number for the contents of this configuration file.
// The value should generally not be modified.
//
version = "V6.0";

////////////////////////////////////////////////////////////////////////////////
//
// Settings specific to individual tools
//
////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////
//
// TCPairsConfig_default
//
////////////////////////////////////////////////////////////////////////////////

//
// The "model" entry specifies an array of model names to be verified. If
// verifying multiple models, choose descriptive model names (no whitespace)
// to distinguish between their output.
// e.g. model = [ "AHW4", "AHWI" ];
//
model  = [];

//
// Specify whether the code should check for duplicate ATCF lines when
// building tracks.  Setting this to FALSE makes the parsing of tracks quicker.
//
// e.g. check_dup = FALSE;
//
check_dup = FALSE;

//
// Specify whether special processing should be performed for interpolated model
// names ending in 'I' (e.g. AHWI).  Search for corresponding tracks whose model
// name ends in '2' (e.g. AHW2) and apply the following logic:
//  - "NONE"    to do nothing.
//  - "FILL"    to create a copy of '2' track and rename it as 'I' only when the
//              'I' track does not already exist.
//  - "REPLACE" to create a copy of the '2' track and rename it as 'I' in all
```

```
//              cases, replacing any 'I' tracks that may already exist.
//
interp12 = REPLACE;

//
// Specify how consensus forecasts should be defined:
//   name    = consensus model name
//   members = array of consensus member model names
//   required = array of TRUE/FALSE for each member
//              if empty, default is FALSE
//   min_req = minimum number of members required for the consensus
//
// e.g.
//    consensus = [
//       {
//          name     = "CON1";
//          members  = [ "MOD1", "MOD2", "MOD3" ];
//          required = [ TRUE, FALSE, FALSE ];
//          min_req  = 2;
//       }
//    ];
//
consensus = [];

//
// Specify a comma-separated list of forecast lag times to be used in HH[MMSS]
// format.  For each ADECK track identified, a lagged track will be derived
// for each entry listed.
//
// e.g. lag_time = [ "06", "12" ];
//
lag_time = [];

//
// Specify comma-separated lists of CLIPER/SHIFOR baseline forecasts to be
// derived from the BEST and operational tracks, as defined by the
// best_technique and oper_technique settings.
//
// Derived from BEST tracks: BCLP, BCS5, BCD5, BCLA
// Derived from OPER tracks: OCLP, OCS5, OCD5, OCDT
//
// e.g. best_technique = [ "BEST" ];
//      base_baseline  = [ "BCLP", "BCS5", "BCD5", "BCLA" ];
```

```
//      oper_technique = [ "CARQ" ];
//      oper_baseline  = [ "OCLP", "OCS5", "OCD5", "OCDT" ];
//
best_technique = [ "BEST" ];
best_baseline  = [];
oper_technique = [ "CARQ" ];
oper_baseline  = [];

//
// Analysis tracks consist of multiple track points with a lead time of zero
// for the same storm. An analysis track may be generated by running model
// analysis fields through a tracking algorithm. Specify which datasets should
// be searched for analysis track data by setting this to NONE, ADECK, BDECK,
// or BOTH. Use BOTH to create pairs using two different analysis tracks.
//
// e.g. anly_track = BDECK;
//
anly_track = BDECK;

//
// Specify whether only those track points common to both the ADECK and BDECK
// tracks should be written out.
//
// e.g. match_points = FALSE;
//
match_points = FALSE;

//
// Specify the NetCDF output of the gen_dland tool containing a gridded
// representation of the minimum distance to land.
//
dland_file = "MET_BASE/tc_data/dland_nw_hem_tenth_degree.nc";

//
// Specify watch/warning information.  Specify an ASCII file containing
// watch/warning information to be used.  At each track point, the most severe
// watch/warning status in effect, if any, will be written to the output.
// Also specify a time offset in seconds to be added to each watch/warning
// time processed.  NHC applies watch/warning information to all track points
// occurring 4 hours (-14400 second) prior to the watch/warning time.
//
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
```

```
   time_offset = -14400;
}

////////////////////////////////////////////////////////////////////////////////
//
// TCStatConfig_default
//
////////////////////////////////////////////////////////////////////////////////

//
// Stratify by the AMODEL or BMODEL columns.
// Specify comma-separated lists of model names to be used for all analyses
// performed.  May add to this list using the "-amodel" and "-bmodel"
// job command options.
// e.g. amodel = [ "AHW4" ];
//      bmodel = [ "BEST" ];
//
amodel = [];
bmodel = [];

//
// Stratify by the VALID times.
// Define beginning and ending time windows in YYYYMMDD[_HH[MMSS]]
// or provide a list of specific valid times to include or exclude.
// May modify using the "-valid_beg", "-valid_end", "-valid_inc",
// and "-valid_exc" job command options.
//
// e.g. valid_beg = "20100101";
//      valid_end = "20101231_12";
//      valid_inc = [ "20101231_06" ];
//      valid_exc = [ "20101231_00" ];
//
valid_beg = "";
valid_end = "";
valid_inc = [];
valid_exc = [];

//
// Stratify by the initialization and valid hours and lead time.
// Specify a comma-separated list of initialization hours,
// valid hours, and lead times in HH[MMSS] format.
// May add using the "-init_hour", "-valid_hour", "-lead",
// and "-lead_req" job command options.
```

```
//
// e.g. init_hour  = [ "00" ];
//      valid_hour = [ "12" ];
//      lead       = [ "24", "36" ];
//      lead_req   = [ "72", "84", "96", "108" ];
//
init_hour  = [];
valid_hour = [];
lead       = [];
lead_req   = [];

//
// Stratify by the LINE_TYPE column.  May add using the "-line_type"
// job command option.
//
// e.g. line_type = [ "TCMPR" ];
//
line_type = [];

//
// Stratify by checking the watch/warning status for each track point
// common to both the ADECK and BDECK tracks. If the watch/warning status
// of any of the track points appears in the list, retain the entire track.
// Individual watch/warning status by point may be specified using the
// -column_str options below, but this option filters by the track maximum.
// May add using the "-track_watch_warn" job command option.
// The value "ALL" matches HUWARN, TSWARN, HUWATCH, and TSWATCH.
//
// e.g. track_watch_warn = [ "HUWATCH", "HUWARN" ];
//
track_watch_warn = [];

//
// Stratify by applying thresholds to numeric data columns.
// Specify a comma-separated list of columns names and thresholds
// to be applied.  May add using the "-column_thresh name thresh" job command
// options.
//
// e.g. column_thresh_name = [ "ADLAND", "BDLAND" ];
//      column_thresh_val  = [ >200,     >200     ];
//
column_thresh_name = [];
column_thresh_val  = [];
```

```
//
// Stratify by performing string matching on non-numeric data columns.
// Specify a comma-separated list of columns names and values
// to be checked.  May add using the "-column_str name string" job command
// options.
//
// e.g. column_str_name = [ "LEVEL", "LEVEL" ];
//      column_str_val  = [ "HU",    "TS"    ];
//
column_str_name = [];
column_str_val  = [];

//
// Just like the column_thresh options above, but apply the threshold only
// when lead = 0.  If lead = 0 value does not meet the threshold, discard
// the entire track.  May add using the "-init_thresh name thresh" job command
// options.
//
// e.g. init_thresh_name = [ "ADLAND" ];
//      init_thresh_val  = [ >200     ];
//
init_thresh_name = [];
init_thresh_val  = [];

//
// Just like the column_str options above, but apply the string matching only
// when lead = 0.  If lead = 0 string does not match, discard the entire track.
// May add using the "-init_str name thresh" job command options.
//
// e.g. init_str_name = [ "LEVEL" ];
//      init_str_val  = [ "HU"    ];
//
init_str_name = [];
init_str_val  = [];

//
// Stratify by the ADECK and BDECK distances to land.  Once either the ADECK or
// BDECK track encounters land, discard the remainder of the track.
//
// e.g. water_only = FALSE;
//
water_only = FALSE;
```

```
//
// Specify whether only those track points for which rapid intensification
// or weakening of the maximum wind speed occurred in the previous time
// step should be retained.
//
// The NHC considers a 24-hour change >=30 kts to constitute rapid
// intensification or weakening.
//
// May modify using the following job command options:
//    "-rirw_track"
//    "-rirw_time" for both or "-rirw_time_adeck" and "-rirw_time_bdeck"
//    "-rirw_exact" for both or "-rirw_exact_adeck" and "-rirw_exact_bdeck"
//    "-rirw_thresh" for both or "-rirw_thresh_adeck" and "-rirw_thresh_bdeck"
//

rirw = {
   track  = NONE;       // Specify which track types to search (NONE, ADECK,
                        // BDECK, or BOTH)
   adeck = {
      time   = "24";    // Rapid intensification/weakening time period in HHMMSS
                        // format.
      exact  = TRUE;    // Use the exact or maximum intensity difference over the
                        // time period.
      thresh = >=30.0;  // Threshold for the intensity change.
   }
   bdeck = adeck;       // Copy settings to the BDECK or specify different logic.
}

//
// Specify whether only those track points occurring near landfall should be
// retained, and define the landfall retention window as a timestring in HH[MMSS]
// format (or as an integer number of seconds) offset from the landfall time.
// Landfall is defined as the last BDECK track point before the distance to land
// switches from positive to 0 or negative.
//
// May modify using the "-landfall_window" job command option, which
// automatically sets -landfall to TRUE.
//
// The "-landfall_window" job command option takes 1 or 2 arguments in  HH[MMSS]
// format.  Use 1 argument to define a symmetric time window.  For example,
// "-landfall_window 06" defines the time window +/- 6 hours around the landfall
// time.  Use 2 arguments to define an asymmetric time window.  For example,
```

```
// "-landfall_window 00 12" defines the time window from the landfall event to 12
// hours after.
//
// e.g. landfall     = FALSE;
//      landfall_beg = "-24"; (24 hours prior to landfall)
//      landfall_end = "00";
//
landfall     = FALSE;
landfall_beg = "-24";
landfall_end = "00";

//
// Specify whether only those cases common to all models in the dataset should
// be retained.  May modify using the "-event_equal" job command option.
//
// e.g. event_equal = FALSE;
//
event_equal = FALSE;

//
// Specify lead times that must be present for a track to be included in the
// event equalization logic.
//
event_equal_lead = [ "12", "24", "36" ];

//
// Apply polyline masking logic to the location of the ADECK track at the
// initialization time.  If it falls outside the mask, discard the entire track.
// May modify using the "-out_init_mask" job command option.
//
// e.g. out_init_mask = "";
//
out_init_mask = "";

//
// Apply polyline masking logic to the location of the ADECK track at the
// valid time.  If it falls outside the mask, discard only the current track
// point.  May modify using the "-out_valid_mask" job command option.
//
// e.g. out_valid_mask = "";
//
out_valid_mask = "";
```

```
//
// The "jobs" entry is an array of TCStat jobs to be performed.
// Each element in the array contains the specifications for a single analysis
// job to be performed.  The format for an analysis job is as follows:
//
//    -job job_name
//    OPTIONAL ARGS
//
//    Where "job_name" is set to one of the following:
//
//       "filter"
//          To filter out the TCST lines matching the job filtering criteria
//          specified above and using the optional arguments below.  The
//          output TCST lines are written to the file specified using the
//          "-dump_row" argument.
//          Required Args: -dump_row
//
//          To further refine the TCST data: Each optional argument may be used
//          in the job specification multiple times unless otherwise indicated.
//          When multiple optional arguments of the same type are indicated, the
//          analysis will be performed over their union
//
//          "-amodel            name"
//          "-bmodel            name"
//          "-lead        HHMMSS"
//          "-valid_beg   YYYYMMDD[_HH[MMSS]]" (use once)
//          "-valid_end   YYYYMMDD[_HH[MMSS]]" (use once)
//          "-valid_inc   YYYYMMDD[_HH[MMSS]]" (use once)
//          "-valid_exc   YYYYMMDD[_HH[MMSS]]" (use once)
//          "-init_beg    YYYYMMDD[_HH[MMSS]]" (use once)
//          "-init_end    YYYYMMDD[_HH[MMSS]]" (use once)
//          "-init_inc    YYYYMMDD[_HH[MMSS]]" (use once)
//          "-init_exc    YYYYMMDD[_HH[MMSS]]" (use once)
//          "-init_hour   HH[MMSS]"
//          "-valid_hour  HH[MMSS]
//          "-init_mask          name"
//          "-valid_mask         name"
//          "-line_type          name"
//          "-track_watch_warn   name"
//          "-column_thresh      name thresh"
//          "-column_str         name string"
//          "-init_thresh        name thresh"
//          "-init_str           name string"
```

```
//
//          Additional filtering options that may be used only when -line_type
//          has been listed only once. These options take two arguments: the name
//          of the data column to be used and the min, max, or exact value for
//          that column. If multiple column eq/min/max/str options are listed,
//          the job will be performed on their intersection:
//
//          "-column_min col_name value"  e.g. -column_min TK_ERR 100.00
//          "-column_max col_name value"
//          "-column_eq  col_name value"
//          "-column_str col_name string" separate multiple filtering strings
//                                        with commas
//
//          Required Args: -dump_row
//
//       "summary"
//          To compute the mean, standard deviation, and percentiles
//          (0th, 10th, 25th, 50th, 75th, 90th, and 100th) for the statistic
//          specified using the "-line_type" and "-column" arguments.
//          For TCStat, the "-column" argument may be set to:
//
//             "TRACK" for track, along-track, and cross-track errors.
//             "WIND" for all wind radius errors.
//             "TI" for track and maximum wind intensity errors.
//             "AC" for along-track and cross-track errors.
//             "XY" for x-track and y-track errors.
//             "col" for a specific column name.
//             "col1-col2" for a difference of two columns.
//             "ABS(col or col1-col2)" for the absolute value.
//
//          Use the -column_union TRUE/FALSE job command option to compute
//          summary statistics across the union of input columns rather than
//          processing them separately.
//
//          Required Args: -line_type, -column
//          Optional Args: -by column_name to specify case information
//                         -out_alpha to override default alpha value
//                         -column_union to summarize multiple columns
//
//       "rirw"
//          To define rapid intensification/weakening contingency table using
//          the ADECK and BDECK RI/RW settings and the matching time window
//          and output contingency table counts and statistics.
```

```
//
//          Optional Args:
//             -rirw_window width in HH[MMSS] format to define a symmetric time
//                window
//             -rirw_window beg end in HH[MMSS] format to define an asymmetric
//                time window
//              Default search time window is 0 0, requiring exact match
//             -rirw_time or -rirw_time_adeck and -rirw_time_bdeck to override
//                defaults
//             -rirw_exact or -rirw_exact_adeck and -rirw_exact_bdeck to override
//                defaults
//             -rirw_thresh or -rirw_thresh_adeck and -rirw_thresh_bdeck to
//                override defaults
//             -by column_name to specify case information
//             -out_alpha to override default alpha value
//             -out_line_type to specify output line types (CTC, CTS, and MPR)
//
//          Note that the "-dump_row path" option results in 4 files being
//          created:
//             path_FY_OY.tcst, path_FY_ON.tcst, path_FN_OY.tcst, and
//             path_FN_ON.tcst, containing the TCST lines that were hits, false
//             alarms, misses, and correct negatives,  respectively.  These files
//             may be used as input for additional TC-Stat analysis.
//
//       "probrirw"
//          To define an Nx2 probabilistic contingency table by reading the
//          PROBRIRW line type, binning the forecast probabilities, and writing
//          output probabilistic counts and statistics.
//
//          Required Args:
//             -probrirw_thresh to define the forecast probabilities to be
//                evaluated (e.g. -probrirw_thresh 30)
//
//          Optional Args:
//             -probrirw_exact TRUE/FALSE to verify against the exact (e.g.
//                BDELTA column) or maximum (e.g. BDELTA_MAX column) intensity
//                change in the BEST track
//             -probrirw_bdelta_thresh to define BEST track change event
//                threshold (e.g. -probrirw_bdelta_thresh >=30)
//             -probrirw_prob_thresh to define output probability thresholds
//                (e.g. -probrirw_prob_thresh ==0.1)
//             -by column_name to specify case information
//             -out_alpha to override default alpha value
```

```
//             -out_line_type to specify output line types (PCT, PSTD, PRC, and
//                PJC)
//
//       For the PROBRIRW line type, PROBRIRW_PROB is a derived column name.
//       For example, the following options select 30 kt probabilities and match
//       probability values greater than 0:
//         -probrirw_thresh 30 -column_thresh PROBRIRW_PROB >0
//
//       e.g.
//       jobs = [
//          "-job filter -amodel AHW4 -dumprow ./tc_filter_job.tcst",
//          "-job filter -column_min TK_ERR 100.000 \
//           -dumprow ./tc_filter_job.tcst",
//          "-job summary -line_type TCMPR -column AC \
//           -dumprow  ./tc_summary_job.tcst",
//          "-job rirw -amodel AHW4 -dump_row ./tc_rirw_job" ]
//
jobs = [];

////////////////////////////////////////////////////////////////////////////////
//
// TCGenConfig_default
//
////////////////////////////////////////////////////////////////////////////////

//
// Model initialization frequency in hours, starting at 0.
//
init_freq = 6;

//
// Lead times in hours to be searched for genesis events.
//
lead_window = {
   beg = 24;
   end = 120;
}

//
// Minimum track duration for genesis event in hours.
//
min_duration = 12;
```

```
//
// 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 = {
   vmax_thresh = NA;
   mslp_thresh = NA;
}

//
// 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 = {
   technique   = "BEST";
   category    = [ "TD", "TS" ];
   vmax_thresh = NA;
   mslp_thresh = NA;
}

//
// Operational track genesis event criteria.  Defined as tracks reaching the
// specified intensity category, maximum wind speed threshold, and minimum
// sea-level pressure threshold.  The operational track genesis time is valid
// time of the first track point where all of these criteria are met.
//
oper_genesis = {
   technique   = "CARQ";
   category    = [ "DB", "LO", "WV" ];
   vmax_thresh = NA;
   mslp_thresh = NA;
}

////////////////////////////////////////////////////////////////////////////////
//
// Track filtering options which may be specified separately in each filter
// array entry.
//
////////////////////////////////////////////////////////////////////////////////
```

```
//
// Filter is an array of dictionaries containing the track filtering options
// listed below.  If empty, a single filter is defined using the top-level
// settings.
//
filter = [];

//
// Description written to output DESC column
//
desc = "NA";

//
// Forecast ATCF ID's
// If empty, all ATCF ID's found will be processed.
// Statistics will be generated separately for each ATCF ID.
//
model = [];

//
// BEST and operational track storm identifiers
//
storm_id = [];

//
// BEST and operational track storm names
//
storm_name = [];

//
// Forecast and operational initialization time window
//
init_beg = "";
init_end = "";

//
// Forecast, BEST, and operational valid time window
//
valid_beg = "";
valid_end = "";

//
```

```
// Forecast and operational initialization hours
//
init_hour = [];

//
// Forecast and operational lead times in hours
//
lead = [];

//
// Spatial masking region (path to gridded data file or polyline file)
//
vx_mask = "";

//
// Distance to land threshold
//
dland_thresh = NA;

//
// Genesis matching time window, in hours relative to the forecast genesis time
//
genesis_window = {
   beg = -24;
   end =  24;
}

//
// Genesis matching search radius in km.
//
genesis_radius = 300;

////////////////////////////////////////////////////////////////////////////////
//
// Global settings.
//
////////////////////////////////////////////////////////////////////////////////

//
// Confidence interval alpha value
//
ci_alpha = 0.05;
```

```
//
// Statistical output types
//
output_flag = {
   fho    = NONE;
   ctc    = BOTH;
   cts    = BOTH;
}
```

Chapter 4

Re-Formatting of Point Observations

There are several formats of point observations that may preprocessed using the suite of reformatting tools in MET. These include PrepBUFR data from NCEP, SURFRAD data from NOAA, AERONET data from NASA, MADIS data from NOAA, little_r from WRF simulations, and user-defined data in a generic ASCII format. These steps are represented by the first columns in the MET flowchart depicted in Figure 1.1. The software tools used to reformat point data are described in this chapter.

4.1 PB2NC tool

This section describes how to configure and run the PB2NC tool. The PB2NC tool is used to stratify the contents of an input PrepBUFR point observation file and reformat it into NetCDF format for use by other MET tools. The PB2NC tool must be run on the input PrepBUFR point observation file prior to performing verification with the MET statistics tools.

4.1.1 pb2nc usage

The usage statement for the PB2NC tool is shown below:

```
Usage: pb2nc
       prepbufr_file
       netcdf_file
       config_file
       [-pbfile PrepBUFR_file]
       [-valid_beg time]
       [-valid_end time]
       [-nmsg n]
```

```
[-dump path]
[-index]
[-log file]
[-v level]
[-compress level]
```

pb2nc has both required and optional arguments.

Required arguments for pb2nc

1. The **prepbufr_file** argument is the input PrepBUFR file to be processed.

2. The **netcdf_file** argument is the output NetCDF file to be written.

3. The **config_file** argument is the configuration file to be used. The contents of the configuration file are discussed below.

Optional arguments for pb2nc

1. The **-pbfile prepbufr_file** option is used to pass additional input PrepBUFR files.

2. The **-valid_beg time** option in YYYYMMDD[_HH[MMSS]] format sets the beginning of the retention time window.

3. The **-valid_end time** option in YYYYMMDD[_HH[MMSS]] format sets the end of the retention time window.

4. The **-nmsg num_messages** option may be used for testing purposes. This argument indicates that only the first "num_messages" PrepBUFR messages should be processed rather than the whole file. This option is provided to speed up testing because running the PB2NC tool can take a few minutes for each file. Most users will not need this option.

5. The **-dump path** option may be used to dump the entire contents of the PrepBUFR file to several ASCII files written to the directory specified by "path". The user may use this option to view a human-readable version of the input PrepBUFR file, although writing the contents to ASCII files can be slow.

6. The **-index** option shows the available variables with valid data from the BUFR input. It collects the available variable list from BUFR input and checks the existence of valid data and directs the variable names with valid data to the screen. The NetCDF output won't be generated.

7. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

8. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

9. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

An example of the pb2nc calling sequence is shown below:

```
pb2nc sample_pb.blk \
sample_pb.nc \
PB2NCConfig
```

In this example, the PB2NC tool will process the input **sample_pb.blk** file applying the configuration specified in the **PB2NCConfig** file and write the output to a file named **sample_pb.nc**.

4.1.2 pb2nc configuration file

The default configuration file for the PB2NC tool named **PB2NCConfig_default** can be found in the installed share/met/config directory. The version used for the example run in Section 2.10 is available in **scripts/config**. It is recommended that users make a copy of configuration files prior to modifying their contents.

When editing configuration files, environment variables may be used for setting the configurable parameters if convenient. The configuration file parser expands any environment variables to their full value before proceeding. Within the configuration file, environment variables must be specified in the form: **${VAR_NAME}**.

For example, using an environment variable to set the **message_type** (see below) parameter to use ADPUPA and ADPSFC message types might consist of the following:

* In a C-Shell: **setenv MSG_TYP ' "ADPUPA", "ADPSFC" '**

* In the configuration file: **message_type[] = [${MSG_TYP}];**

The contents of the default pb2nc configuration file are described below.

```
obs_window = { beg  = -5400; end  = 5400; }
mask       = { grid = "";    poly = "";   }
tmp_dir    = "/tmp";
version    = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
message_type = [];
```

Each PrepBUFR message is tagged with one of eighteen message types as listed in the share/met/config/README file. The 'message_type' refers to the type of observation from which the observation value (or 'report') was derived. The user may specify a comma-separated list of message types to be retained. Providing an empty list indicates that all message types should be retained.

```
message_type_map = [ { key = ‘‘AIRCAR’’; val = ‘‘AIRCAR_PROFILES’’; } ];
```

The **message_type_map** entry is an array of dictionaries, each containing a **key** string and **val** string. This defines a mapping of input PrepBUFR message types to output message types. This provides a method for renaming input PrepBUFR 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";                     }
];
```

The **message_type_group_map** entry is an array of dictionaries, each containing a **key** string and **val** string. This defines a mapping of message type group names to a comma-separated list of values. This map is defined in the config files for PB2NC, Point-Stat, or Ensemble-Stat. Modify this map to define sets of message types that should be processed together as a group. The **SURFACE** entry must be present to define message types for which surface verification logic should be applied.

```
station_id = [];
```

Each PrepBUFR message has a station identification string associated with it. The user may specify a comma-separated list of station IDs to be retained. Providing an empty list indicates that messages from all station IDs will be retained. It can be a file name containing a list of stations.

```
elevation_range = { beg = -1000; end = 100000; }
```

The **beg** and **end** variables are used to stratify the elevation (in meters) of the observations to be retained. The range shown above is set to -1000 to 100000 meters, which essentially retains every observation.

```
pb_report_type  = [];
in_report_type  = [];
instrument_type = [];
```

The **pb_report_type, in_report_type**, and **instrument_type** variables are used to specify comma-separated lists of PrepBUFR report types, input report types, and instrument types to be retained, respectively. If left empty, all PrepBUFR report types, input report types, and instrument types will be retained. See the following for more details:

http://www.emc.ncep.noaa.gov/mmb/data_processing/PrepBUFR.doc/table_4.htm

http://www.emc.ncep.noaa.gov/mmb/data_processing/PrepBUFR.doc/table_6.htm

---

```
level_range    = { beg = 1; end = 255; }
level_category = [];
```

The **beg** and **end** variables are used to stratify the model level of observations to be retained. The range shown above is 1 to 255.

The **level_category** variable is used to specify a comma-separated list of PrepBUFR data level categories to retain. An empty string indicates that all level categories should be retained. Accepted values and their meanings are described in Table 4.1. See the following for more details:

http://www.emc.ncep.noaa.gov/mmb/data_processing/PrepBUFR.doc/table_1.htm

Table 4.1: Values for the level_category option.

Level category value	Description
0	Surface level
1	Mandatory level
2	Significant temperature level
3	Winds-by-pressure level
4	Winds-by-height level
5	Tropopause level
6	Reports on a single level
7	Auxiliary levels generated via interpolation from spanning levels

```
obs_bufr_var = [ 'QOB', 'TOB', 'ZOB', 'UOB', 'VOB' ];
```

Each PrepBUFR message will likely contain multiple observation variables. The **obs_bufr_var** variable is used to specify which observation variables should be retained or derived. The variable name comes from BUFR file which includes BUFR table. The following BUFR names may be retained: QOB, TOB, ZOB, UOB, and VOB for specific humidity, temperature, height, and the u and v components of winds. The following BUFR names may be derived: D_DPT, D_WIND, D_RH, D_MIXR, D_PRMSL, D_PBL, and D_CAPE for dew point, wind speed, relative humidity, mixing ratio, pressure reduced to MSL, planetary boundary layer height, and convective available potential energy. This configuration replaces **obs_grib_code**. If the list is empty, all BUFR variables are retained.

```
obs_bufr_map = [

    { key = 'POB';      val = 'PRES';  },
    { key = 'QOB';      val = 'SPFH';  },
    { key = 'TOB';      val = 'TMP';   },
    { key = 'ZOB';      val = 'HGT';   },
    { key = 'UOB';      val = 'UGRD';  },
    { key = 'VOB';      val = 'VGRD';  },
    { key = 'D_DPT';    val = 'DPT';   },
    { key = 'D_WDIR';   val = 'WDIR';  },
    { key = 'D_WIND';   val = 'WIND';  },
    { key = 'D_RH';     val = 'RH';    },
    { key = 'D_MIXR';   val = 'MIXR';  },
    { key = 'D_PRMSL';  val = 'PRMSL'; },
    { key = 'D_PBL';    val = 'PBL';   },
    { key = 'D_CAPE';   val = 'CAPE';  }
    ];
```

The BUFR variable names are not shared with other forecast data. This map is used to convert the BUFR name to the common name, like GRIB2. It allows to share the configuration for forecast data with PB2NC observation data. If there is no mapping, the BUFR variable name will be saved to output NetCDF file.

```
quality_mark_thresh = 2;
```

Each observation has a quality mark value associated with it. The **quality_mark_thresh** is used to stratify out which quality marks will be retained. The value shown above indicates that only observations with quality marks less than or equal to 2 will be retained.

```
event_stack_flag = TOP;
```

A PrepBUFR message may contain duplicate observations with different quality mark values. The **event_stack_flag** indicates whether to use the observations at the top of the event stack (observation values have had more quality control processing applied) or the bottom of the event stack (observation values have had no quality control processing applied). The flag value of **TOP** listed above indicates the observations with the most amount of quality control processing should be used, the **BOTTOM** option uses the data closest to raw values.

```
time_summary = {
   flag       = FALSE;
   raw_data   = FALSE;
   beg        = "000000";
   end        = "235959";
   step       = 300;
   width      = 600;
   // width   = { beg = -300; end = 300; }
   grib_code  = [];
   obs_var    = [ "TMP", "WDIR", "RH" ];
   type       = [ "min", "max", "range", "mean", "stdev", "median", "p80" ];
   vld_freq   = 0;
   vld_thresh = 0.0;
}
```

The **time_summary** dictionary enables additional processing for observations with high temporal resolution. The **flag** entry toggles the **time_summary** on (**TRUE**) and off (**FALSE**). If the **raw_data** flag is set to TRUE, then both the individual observation values and the derived time summary value will be written to the output. If FALSE, only the summary values are written. Observations may be summarized across the user specified time period defined by the **beg** and **end** entries in HHMMSS format. The **step** entry defines the time between intervals in seconds. The **width** entry specifies the summary interval in seconds. It may either be set as an integer number of seconds for a centered time interval or a dictionary with beginning and ending time offsets in seconds.

This example listed above does a 10-minute time summary (width = 600;) every 5 minutes (step = 300;) throughout the day (beg = “000000”; end = 235959”;). The first interval will be from 23:55:00 the previous

day through 00:04:59 of the current day. The second interval will be from 0:00:00 through 00:09:59. And so on.

The two **width** settings listed above are equivalent. Both define a centered 10-minute time interval. Use the **beg** and **end** entries to define uncentered time intervals. The following example requests observations for one hour prior:

```
width = { beg = -3600; end = 0; }
```

The summaries will only be calculated for the observations specified in the **grib_code** or **obs_var** entries. The **grib_code** entry is an array of integers while the **obs_var** entries is an array of strings. The supported summaries are **min** (minimum), **max** (maximum), **range**, **mean**, **stdev** (standard deviation), **median** and **p**## (percentile, with the desired percentile value specified in place of ##). If multiple summaries are selected in a single run, a string indicating the summary method applied will be appended to the output message type.

The **vld_freq** and **vld_thresh** entries specify the required ratio of valid data for an output time summary value to be computed. This option is only applied when these entries are set to non-zero values. The **vld_freq** entry specifies the expected frequency of observations in seconds. The width of the time window is divided by this frequency to compute the expected number of observations for the time window. The actual number of valid observations is divided by the expected number to compute the ratio of valid data. An output time summary value will only be written if that ratio is greater than or equal to the **vld_thresh** entry. Detailed information about which observations are excluded is provided at debug level 4.

4.1.3 pb2nc output

Each NetCDF file generated by the PB2NC tool contains the dimensions and variables shown in Tables 4.2 and 4.3.

Table 4.2: NetCDF file dimensions for pb2nc output.

pb2nc NetCDF DIMENSIONS	
NetCDF Dimension	**Description**
mxstr, mxstr2, mxstr3	Maximum string lengths (16, 40, and 80)
nobs	Number of PrepBUFR observations in the file (UNLIMITED)
nhdr, npbhdr	Number of PrepBUFR messages in the file (variable)
nhdr_typ, nhdr_sid, nhdr_vld	Number of unique header message type, station ID, and valid time strings (variable)
nobs_qty	Number of unique quality control strings (variable)
obs_var_num	Number of unique observation variable types (variable)

Table 4.3: NetCDF variables in pb2nc output.

pb2nc NetCDF VARIABLES		
NetCDF Variable	**Dimension**	**Description**
obs_qty	nobs	Integer value of the n_obs_qty dimension for the observation quality control string.
obs_hid	nobs	Integer value of the nhdr dimension for the header arrays with which this observation is associated.
obs_vid	nobs	Integer value of the obs_var_num dimension for the observation variable name, units, and description.
obs_lvl	nobs	Floating point pressure level in hPa or accumulation interval.
obs_hgt	nobs	Floating point height in meters above sea level.
obs_val	nobs	Floating point observation value.
hdr_typ	nhdr	Integer value of the nhdr_typ dimension for the message type string.
hdr_sid	nhdr	Integer value of the nhdr_sid dimension for the station ID string.
hdr_vld	nhdr	Integer value of the nhdr_vld dimension for the valid time string.
hdr_lat, hdr_lon	nhdr	Floating point latitude in degrees north and longitude in degrees east.
hdr_elv	nhdr	Floating point elevation of observing station in meters above sea level.
hdr_prpt_typ	npbhdr	Integer PrepBUFR report type value.
hdr_irpt_typ	npbhdr	Integer input report type value.
hdr_inst_typ	npbhdr	Integer instrument type value.
hdr_typ_table	nhdr_typ, mxstr2	Lookup table containing unique message type strings.
hdr_sid_table	nhdr_sid, mxstr2	Lookup table containing unique station ID strings.
hdr_vld_table	nhdr_vld, mxstr	Lookup table containing unique valid time strings in YYYYMMDD_HHMMSS UTC format.
obs_qty_table	nobs_qty, mxstr	Lookup table containing unique quality control strings.
obs_var	obs_var_num, mxstr	Lookup table containing unique observation variable names.
obs_unit	obs_var_num, mxstr2	Lookup table containing a units string for the unique observation variable names in obs_var.
obs_desc	obs_var_num, mxstr3	Lookup table containing a description string for the unique observation variable names in obs_var.

4.2 ASCII2NC tool

This section describes how to run the ASCII2NC tool. The ASCII2NC tool is used to reformat ASCII point observations into the NetCDF format expected by the Point-Stat tool. For those users wishing to verify against point observations that are not available in PrepBUFR format, the ASCII2NC tool provides a way of incorporating those observations into MET. If the ASCII2NC tool is used to perform a reformatting step,

no configuration file is needed. However, for more complex processing, such as summarizing time series observations, a configuration file may be specified. For details on the configuration file options, see the share/met/config/README file and example configuration files distributed with the MET code.

Initial versions of the ASCII2NC tool supported only a simple 11 column ASCII point observation format. It currently supports point observation data in the following formats: the default 11 column format, little_r format, SURFace RADiation (SURFRAD) and Integrated Surface Irradiance Study (ISIS) formats (found at `http://www.esrl.noaa.gov/gmd/grad/surfrad/`), the Western Wind and Solar Integration Study (WWSIS) format, and the AErosol RObotic NEtwork (AERONET) versions 2 and 3 format (found at `http://aeronet.gsfc.nasa.gov/`). WWSIS data are available by request from National Renewable Energy Laboratory (NREL) in Boulder, CO.

MET version 9.0 adds support for the passing observations to ascii2nc using a Python script with the "-format python" option. An example of running ASCII2NC with Python embedding is included below.

The default ASCII point observation format consists of one row of data per observation value. Each row of data consists of 11 columns as shown in Table 4.4.

Table 4.4: Input MET ascii2nc point observation format

ascii2nc ASCII Point Observation Format		
Column	**Name**	**Description**
1	Message_Type	Text string containing the observation message type as described in the previous section on the PB2NC tool.
2	Station_ID	Text string containing the station id.
3	Valid_Time	Text string containing the observation valid time in YYYYMMDD_HHMMSS format.
4	Lat	Latitude in degrees north of the observing location.
5	Lon	Longitude in degrees east of the observation location.
6	Elevation	Elevation in msl of the observing location.
7	GRIB_Code or Variable_Name	Integer GRIB code value or variable name corresponding to this observation type.
8	Level	Pressure level in hPa or accumulation interval in hours for the observation value.
9	Height	Height in msl or agl of the observation value.
10	QC_String	Quality control value.
11	Observation_Value	Observation value in units consistent with the GRIB code definition.

4.2.1 ascii2nc usage

Once the ASCII point observations have been formatted as expected, the ASCII file is ready to be processed by the ASCII2NC tool. The usage statement for ASCII2NC tool is shown below:

```
Usage: ascii2nc
```

```
ascii_file1 [ascii_file2 ... ascii_filen]
netcdf_file
[-format ASCII_format]
[-config file]
[-mask_grid string]
[-mask_poly file]
[-mask_sid file|list]
[-log file]
[-v level]
[-compress level]
```

ascii2nc has two required arguments and can take several optional ones.

Required arguments for ascii2nc

1. The **ascii_file** argument is the ASCII point observation file(s) to be processed. If using Python embedding with "-format python" provide a quoted string containing the Python script to be run followed by any command line arguments that script takes.

2. The **netcdf_file** argument is the NetCDF output file to be written.

Optional arguments for ascii2nc

3. The **-format ASCII_format** option may be set to "met_point", "little_r", "surfrad", "wwsis", "aeronet", "aeronetv2", "aeronetv3", or "python". If passing in ISIS data, use the "surfrad" format flag.

4. The **-config file** option is the configuration file for generating time summaries.

5. The **-mask_grid string** option is a named grid or a gridded data file to filter the point observations spatially.

6. The **-mask_poly file** option is a polyline masking file to filter the point observations spatially.

7. The **-mask_sid file|list** option is a station ID masking file or a comma-separated list of station ID's to filter the point observations spatially. See the description of the "sid" entry in 3.5.1.

8. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

9. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

10. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

An example of the **ascii2nc** calling sequence is shown below:

```
ascii2nc sample_ascii_obs.txt \
sample_ascii_obs.nc
```

In this example, the ASCII2NC tool will reformat the input **sample_ascii_obs.txt file** into NetCDF format and write the output to a file named **sample_ascii_obs.nc**.

4.2.1.1 Python Embedding for Point Observations

Here is an example of processing the same set of observations but using Python embedding instead:

```
ascii2nc -format python \
"MET_BASE/python/read_ascii_point.py sample_ascii_obs.txt" \
sample_ascii_obs_python.nc
```

Please refer to Appendix F for more details about Python embedding in MET.

4.2.2 ascii2nc configuration file

The default configuration file for the ASCII2NC tool named **Ascii2NcConfig_default** can be found in the installed share/met/config directory. It is recommended that users make a copy of this file prior to modifying its contents.

The ASCII2NC configuration file is optional and only necessary when defining time summaries or message type mapping for little_r data. The contents of the default ASCII2NC configuration file are described below.

```
version = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
time_summary = { ... }
```

The **time_summary** feature was implemented to allow additional processing of observations with high temporal resolution, such as SURFRAD data every 5 minutes. This option is described in Section 4.1.2.

```
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"; }
];
```

This entry is an array of dictionaries, each containing a **key** string and **val** string which define a mapping of input strings to output message types. This mapping is currently only applied when converting input little_r report types to output message types.

4.2.3 ascii2nc output

The NetCDF output of the ASCII2NC tool is structured in the same way as the output of the PB2NC tool described in Section 4.1.3.

4.3 MADIS2NC tool

This section describes how to run the MADIS2NC tool. The MADIS2NC tool is used to reformat Meteorological Assimilation Data Ingest System (MADIS) point observations into the NetCDF format expected by the MET statistics tools. More information about MADIS data and formatting is available at `http://madis.noaa.gov`. Since the MADIS2NC tool simply performs a reformatting step, no configuration file is needed. The MADIS2NC tool supports many of the MADIS data types, as listed in the usage statement below. Support for additional MADIS data types may be added in the future based on user feedback.

4.3.1 madis2nc usage

The usage statement for MADIS2NC tool is shown below:

```
Usage: madis2nc
       madis_file [madis_file2 ... madis_filen]
       out_file
       -type str
       [-config file]
       [-qc_dd list]
       [-lvl_dim list]
       [-rec_beg n]
       [-rec_end n]
       [-mask_grid string]
       [-mask_poly file]
       [-mask_sid file|list]
       [-log file]
       [-v level]
       [-compress level]
```

madis2nc has required arguments and can also take optional ones.

Required arguments for madis2nc

1. The **madis_file** argument is one or more input MADIS point observation files to be processed.
2. The **netcdf_file** argument is the NetCDF output file to be written.
3. The argument **-type str** is type of MADIS observations (metar, raob, profiler, maritime, mesonet or acarsProfiles).

Optional arguments for madis2nc

4. The **-config file** option specifies the configuration file to generate summaries of the fields in the ASCII files.
5. The **-qc_dd list** option specifies a comma-separated list of QC flag values to be accepted (Z,C,S,V,X,Q,K,G,B).
6. The **-lvl_dim list option** specifies a comma-separated list of vertical level dimensions to be processed.
7. To specify the exact records to be processed, the **-rec_beg n** specifies the index of the first MADIS record to process and **-rec_end n** specifies the index of the last MADIS record to process. Both are zero-based.

8. The **-mask_grid string** option specifies a named grid or a gridded data file for filtering the point observations spatially.

9. The **-mask_poly file** option defines a polyline masking file for filtering the point observations spatially.

10. The **-mask_sid file|list** option is a station ID masking file or a comma-separated list of station ID's for filtering the point observations spatially. See the description of the "sid" entry in 3.5.1.

11. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

12. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity will increase the amount of logging.

13. The **-compress level** option specifies the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

An example of the madis2nc calling sequence is shown below:

```
madis2nc sample_madis_obs.nc \
sample_madis_obs_met.nc -log madis.log -v 3
```

In this example, the MADIS2NC tool will reformat the input sample_madis_obs.nc file into NetCDF format and write the output to a file named sample_madis_obs_met.nc. Warnings and error messages will be written to the madis.log file, and the verbosity level of logging is three.

4.3.2 madis2nc configuration file

The default configuration file for the MADIS2NC tool named **Madis2NcConfig_default** can be found in the installed share/met/config directory. It is recommended that users make a copy of this file prior to modifying its contents.

The MADIS2NC configuration file is optional and only necessary when defining time summaries. The contents of the default MADIS2NC configuration file are described below.

```
version = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
time_summary = { ... }
```

The **time_summary** dictionary is described in Section 4.1.2.

4.3.3 madis2nc output

The NetCDF output of the MADIS2NC tool is structured in the same way as the output of the PB2NC tool described in Section 4.1.3.

4.4 LIDAR2NC tool

The LIDAR2NC tool creates a NetCDF point observation file from a CALIPSO HDF data file. Not all of the data present in the CALIPSO file is reproduced in the output, however. Instead, the output focuses mostly on information about clouds (as opposed to aerosols) as seen by the satellite along its ground track.

4.4.1 lidar2nc usage

The usage statement for LIDAR2NC tool is shown below:

```
Usage: lidar2nc
       lidar_file
       -out out_file
       [-log file]
       [-v level]
       [-compress level]
```

Unlike most of the MET tools, lidar2nc does not use a config file. Currently, the options needed to run lidar2nc are not complex enough to require one.

Required arguments for lidar2nc

1. The **lidar_file** argument is the input HDF lidar data file to be processed. Currently, CALIPSO files are supported but support for additional file types will be added in future releases.

2. The **out_file** argument is the NetCDF output file to be written.

Optional arguments for lidar2nc

3. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

4. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

5. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

4.4.2 lidar2nc output

Each observation type in the lidar2nc output is assigned a GRIB code. These are outlined in Table 4.5. GRIB codes were assigned to these fields arbitrarily, with GRIB codes in the 600s denoting individual bit fields taken from the feature classification flag field in the CALIPSO file.

We will not give a detailed description of each CALIPSO data product that lidar2nc reads. Users should refer to existing CALIPSO documentation for this information. We will, however, give some explanation of how the cloud layer base and top information is encoded in the lidar2nc NetCDF output file.

Layer_Base gives the elevation in meters above ground level of the cloud base for each cloud level at each observation location. Similarly, **Layer_Top** gives the elevation of the top of each cloud layer. Note that if there are multiple cloud layers at a particular location, then there will be more than one base (or top) given for that location. For convenience, **Min_Base** and **Max_Top** give, respectively, the base elevation for the bottom cloud layer, and the top elevation for the top cloud layer. For these data types, there will be only one value per observation location regardless of how many cloud layers there are at that location.

Table 4.5: lidar2nc GRIB codes and their meaning, units, and abbreviations

GRIB Code	Meaning	Units	Abbreviation
500	Number of Cloud Layers	NA	NLayers
501	Cloud Layer Base AGL	m	Layer_Base
502	Cloud Layer Top AGL	m	Layer_Top
503	Cloud Opacity	%	Opacity
504	CAD Score	NA	CAD_Score
505	Minimum Cloud Base AGL	m	Min_Base
506	Maximum Cloud Top AGL	m	Max_Top
600	Feature Type	NA	Feature_Type
601	Ice/Water Phase	NA	Ice_Water_Phase
602	Feature Sub-Type	NA	Feature_Sub_Type
603	Cloud/Aerosol/PSC Type QA	NA	Cloud_Aerosol_PSC_Type_QA
604	Horizontal Averaging	NA	Horizontal_Averaging

4.5 Point2Grid tool

The Point2Grid tool takes point observations from a NetCDF output file from one of the four previously mentioned MET tools (ascii2nc, madis2nc, pb2nc, lidar2nc) and creates a gridded NetCDF file. The other point observations are GOES-16/17 input files in NetCDF format (especially, Aerosol Optical Depth. Future development will include support for reading input files not produced from MET tools.

4.5.1 point2grid usage

The usage statement for the Point2Grid tool is shown below:

```
Usage: point2grid
       input_filename
       to_grid
       output_filename
       -field string
       [-config file]
       [-qc flags]
       [-adp adp_file_name]
       [-method type]
       [-gaussian_dx n]
       [-gaussian_radius n]
       [-prob_cat_thresh n]
       [-vld_thresh n]
       [-name list]
       [-log file]
       [-v level]
       [-compress level]
```

Required arguments for point2grid

1. The **input_filename** argument indicates the name of the input NetCDF file to be processed. Currently, only NetCDF files produced from the ascii2nc, madis2nc, pb2nc, and lidar2nc are supported. And AOD dataset from GOES16/17 are supported, too. Support for additional file types will be added in future releases.

2. The to_grid argument defines the output grid as: (1) a named grid, (2) the path to a gridded data file, or (3) an explicit grid specification string.

3. The output_filename argument is the name of the output NetCDF file to be written.

4. The -field string argument is a string that defines the data to be regridded. It may be used multiple times. If -adp option is given (for AOD data from GOES16/17), the name consists with the variable name from the input data file and the variable name from ADP data file (for example, "AOD_Smoke" or "AOD_Dust": getting AOD variable from the input data and applying smoke or dust variable from ADP data file).

Optional arguments for point2grid

5. The **-config** file option is the configuration file to be used.

6. The **-qc** flags option specifies a comma-separated list of quality control (QC) flags, for example "0,1". This should only be applied if grid_mapping is set to "goes_imager_projection" and the QC variable exists.

7. The **-adp adp_file_name** option provides an additional Aerosol Detection Product (ADP) information on aerosols, dust, and smoke. This option is ignored if the requested variable is not AOD ("AOD_Dust" or "AOD_Smoke") from GOES16/17. The gridded data is filtered by the presence of dust/smoke. If -qc options is given, it's applied to QC of dust/smoke, too (First filtering with AOD QC values and the second filtering with dust/smoke QC values).

8. The **-method type** option specifies the regridding method. The default method is UW_MEAN.

9. The **-gaussian_dx n** option defines the distance interval for Gaussian smoothing. The default is 81.271 km. Ignored if the method is not GAUSSIAN.

10. The **-gaussian_radius n** option defines the radius of influence for Gaussian interpolation. The default is 120. Ignored if the method is not GAUSSIAN.

11. The **-prob_cat_thresh n** option sets the threshold to compute the probability of occurrence. The default is set to disabled. This option is relevant when calculating practically perfect forecasts.

12. The **-vld_thresh n** option sets the required ratio of valid data for regridding. The default is 0.5.

13. The **-name list** option specifies a comma-separated list of output variable names for each field specified.

14. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

15. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

16. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

For the GOES-16 and GOES-17 data, the computing lat/long is time consuming. So the computed coordinate (lat/long) is saved into the NetCDF file to the environment variable MET_TMP_DIR or /tmp if MET_TMP_DIR is not defined. The computing lat/long step can be skipped if the coordinate file is given through the environment variable MET_GEOSTATIONARY_DATA. An example of call point2grid to process GOES-16 AOD data is shown below:

```
point2grid \
OR_ABI-L2-AODC-M3_G16_s20181341702215_e20181341704588_c20181341711418.nc \
G212 \
regrid_data_plane_GOES-16_AOD_TO_G212.nc \
-field 'name="AOD"; level="(*,*)";' \
-qc 0,1,2
-method MAX -v 1
```

When processing GOES-16 data, the **-qc** option may also be used to specify the acceptable quality control flag values. The example above regrids the GOES-16 AOD values to NCEP Grid number 212 (which QC flags are high, medium, and low), writing to the output the maximum AOD value falling inside each grid box.

4.5.2 point2grid output

The point2grid tool will output a gridded NetCDF file containing the following:

1. Latitude

2. Longitude

3. The variable specified in the -field string regridded to the grid defined in the to_grid argument.

4. The count field which represents the number of point observations that were included calculating the value of the variable at that grid cell.

5. The mask field which is a binary field representing the presence or lack thereof of point observations at that grid cell. A value of "1" indicates that there was at least one point observation within the bounds of that grid cell and a value of "0" indicates the lack of point observations at that grid cell.

6. The probability field which is the probability of the event defined by the line option -prob_cat_thresh n occurring. Ranges from 0 to 1.

7. The probability mask field which is a binary field that represents whether or not there is probability data at that grid point. Can be either "0" or "1" with "0" meaning the probability value does not exist and a value of "1" meaning that the probability value does exist.

Chapter 5

Re-Formatting of Gridded Fields

Several MET tools exist for the purpose of reformatting gridded fields, and they are described in this chapter. These tools are represented by the reformatting column of MET flowchart depicted in Figure 1.1.

5.1 Pcp-Combine tool

This section describes the Pcp-Combine tool which summarizes data across multiple input gridded data files and writes the results to a single NetCDF output file. It is often used to modify precipitation accumulation intervals in the forecast and/or observation datasets to make them comparable. However it can also be used to derive summary fields, such as daily min/max temperature or average precipitation rate.

The Pcp-Combine tool supports four types of commands ("sum", "add", "subtract", and "derive") which may be run on any gridded data files supported by MET.

1. The "sum" command is the default command and therefore specifying "-sum" on the command line is optional. Using the sum arguments described below, Pcp-Combine searches the input directories ("-pcpdir" option) for data that matches the requested time stamps and accumulation intervals. Pcp-Combine only considers files from the input data directory which match the specified regular expression ("-pcprx" option). While "sum" searches for matching data, all the other commands are run on the explicit set of input files specified.

2. The "add" command reads the requested data from the input data files and adds them together.

3. The "subtract" command reads the requested data from exactly two input files and computes their difference.

4. The "derive" command reads the requested data from the input data files and computes the requested summary fields.

By default, the Pcp-Combine tool processes data for **APCP**, the GRIB string for accumulated precipitation. When requesting data using time strings (i.e. [HH]MMSS), Pcp-Combine searches for accumulated precipitation for that accumulation interval. Alternatively, use the "-field" option to process fields other than **APCP** or for non-GRIB files. The "-field" option may be used multiple times to process multiple fields in a single run. Since the Pcp-Combine tool does not support automated regridding, all input data must be on the same grid. In general the input files should have the same initialization time unless the user has indicated that it should ignore the initialization time for the "sum" command. The "subtract" command produces a warning when the input initialization times differ or the subtraction results in a negative accumulation interval.

5.1.1 pcp_combine usage

The usage statement for the Pcp-Combine tool is shown below:

```
Usage: pcp_combine
       [-sum] sum_args |
       -add input_files |
       -subtract input_files |
       -derive stat_list input_files
       out_file
       [-field string]
       [-name list]
       [-vld_thresh n]
       [-log file]
       [-v level]
       [-compress level]
```

The arguments to pcp_combine vary depending on the run command. Listed below are the arguments for the sum command:

```
SUM_ARGS:
       init_time
       in_accum
       valid_time
       out_accum
       out_file
       [-pcpdir path]
       [-pcprx reg_exp]
```

The add, subtract, and derive commands all require that the input files be explicitly listed:

```
INPUT_FILES:
       file_1 config_str_1 ... file_n config_str_n |
       file_1 ... file_n |
       input_file_list
```

Required arguments for the pcp_combine

1. The Pcp-Combine tool must be run with exactly one run command (-sum, -add, -subtract, or -derive) with the corresponding additional arguments.

2. The **out_file** argument indicates the name for the NetCDF file to be written.

Optional arguments for pcp_combine

3. The **-field string** option defines the data to be extracted from the input files. Use this option when processing fields other than **APCP** or non-GRIB files. This option may be used multiple times and output will be created for each.

4. The **-name list** option is a comma-separated list of output variable names which override the default choices. If specified, the number of names must match the number of variables to written to the output file.

5. The **-vld_thresh n** option overrides the default required ratio of valid data for at each grid point for an output value to be written. The default is 1.0.

6. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

7. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

8. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

Required arguments for the pcp_combine sum command

1. The **init_time** argument, provided in YYYYMMDD[_HH[MMSS]] format, indicates the initialization time for model data to be summed. Only files found with this initialization time will be processed. If combining observation files, Stage II or Stage IV data for example, the initialization time is not applicable. Providing a string of all zeros (00000000_000000) indicates that all files, regardless of initialization time should be processed.

2. The **in_accum** argument, provided in HH[MMSS] format, indicates the accumulation interval of the model or observation gridded files to be processed. This value must be specified, since a model output file may contain multiple accumulation periods for precipitation in a single file. The argument indicates which accumulation period to extract.

3. The **valid_time** argument, in YYYYMMDD[_HH[MMSS]] format, indicates the desired valid time to which the accumulated precipitation is to be summed.

4. The **out_accum** argument, in HH[MMSS] format, indicates the desired total accumulation period to be summed.

Optional arguments for pcp_combine sum command

5. The **-pcpdir path** option indicates the directories in which the input files reside. The contents of "**path**" will override the default setting. This option may be used multiple times and can accept multiple arguments, supporting the use of wildcards.

6. The **-pcprx reg_exp** option indicates the regular expression to be used in matching files in the search directories specified. The contents of "reg_exp" will override the default setting that matches all file names. If the search directories contain a large number of files, the user may specify that only a subset of those files be processed using a regular expression which will speed up the run time.

Required arguments for the pcp_combine derive command

1. The "derive" run command must be followed by **stat_list** which is a comma-separated list of summary fields to be computed. The **stat_list** may be set to sum, min, max, range, mean, stdev, and vld_count for the sum, minimum, maximum, range (max-min), average, standard deviation, and valid data count fields, respectively.

Input files for pcp_combine add, subtract, and derive commands

The input files for the add, subtract, and derive command can be specified in one of 3 ways:

1. Use **file_1 config_str_1 ... file_n config_str_n** to specify the full path to each input file followed by a description of the data to be read from it. The **config_str_i** argument describing the data can be a set to a time string in HH[MMSS] format for accumulated precipitation or a full configuration string. For example, use **'name="TMP"; level="P500";'** to process temperature at 500mb.

2. Use **file_1 ... file_n** to specify the list of input files to be processed on the command line. Rather then specifying a separate configuration string for each input file, the "-field" command line option is required to specify the data to be processed.

3. Use **input_file_list** to specify the name of an ASCII file which contains the paths for the gridded data files to be processed. As in the previous option, the "-field" command line option is required to specify the data to be processed.

An example of the pcp_combine calling sequence is presented below:

Example 1:

```
pcp_combine -sum \
20050807_000000 3 \
20050808_000000 24 \
sample_fcst.nc \
-pcpdir ../data/sample_fcst/2005080700
```

In Example 1, the Pcp-Combine tool will sum the values in model files initialized at 2005/08/07 00Z and containing 3-hourly accumulation intervals of precipitation. The requested valid time is 2005/08/08 00Z with a requested total accumulation interval of 24 hours. The output file is to be named sample_fcst.nc, and the Pcp-Combine tool is to search the directory indicated for the input files.

The Pcp-Combine tool will search for 8 files containing 3-hourly accumulation intervals which meet the criteria specified. It will write out a single NetCDF file containing that 24 hours of accumulation.

A second example of the pcp_combine calling sequence is presented below:

Example 2:

```
pcp_combine -sum \
00000000_000000 1 \
20050808_000000 24 \
sample_obs.nc \
-pcpdir ../data/sample_obs/ST2ml
```

Example 2 shows an example of using the Pcp-Combine tool to sum observation data. The "init_time" has been set to all zeros to indicate that when searching through the files in precipitation directory, the initialization time should be ignored. The "in_accum" has been changed from 3 to 1 to indicate that the input observation files contain 1-hourly accumulations of precipitation. Lastly, -pcpdir provides a different directory to be searched for the input files.

The Pcp-Combine tool will search for 24 files containing 1-hourly accumulation intervals which meet the criteria specified. It will write out a single NetCDF file containing that 24 hours of accumulation.

Example 3:

```
pcp_combine -add input_pinterp.nc 'name="TT"; level="(0,*,*)";' tt_10.nc
```

This command would grab the first level of the TT variable from a pinterp NetCDF file and write it to the output tt_10.nc file.

5.1.2 pcp_combine output

The output NetCDF files contain the requested accumulation intervals as well as information about the grid on which the data lie. That grid projection information will be parsed out and used by the MET statistics tools in subsequent steps. One may use NetCDF utilities such as ncdump or ncview to view the contents of the output file. Alternatively, the MET Plot-Data-Plane tool described in Section 25.1.2 may be run to create a PostScript image of the data.

Each NetCDF file generated by the Pcp-Combine tool contains the dimensions and variables shown in the following two tables.

Table 5.1: NetCDF file dimensions for pcp_combine output.

Pcp_combine NetCDF dimensions	
NetCDF dimension	**Description**
lat	Dimension of the latitude (i.e. Number of grid points in the North-South direction)
lon	Dimension of the longitude (i.e. Number of grid points in the East-West direction)

Table 5.2: NetCDF variables for pcp_combine output.

Pcp_combine NetCDF variables		
NetCDF variable	**Dimension**	**Description**
lat	lat, lon	Latitude value for each point in the grid
lon	lat, lon	Longitude value for each point in the grid
Name and level of the requested data or value of the -name option.	lat, lon	Data value (i.e. accumulated precipitation) for each point in the grid. The name of the variable describes the name and level and any derivation logic that was applied.

5.2 Regrid_data_plane tool

This section contains a description of running the regrid_data_plane tool. This tool may be run to read data from any gridded file MET supports, interpolate to a user-specified grid, and write the field(s) out in NetCDF format. The user may specify the method of interpolation used for regridding as well as which fields to regrid. This tool is particularly useful when dealing with GRIB2 and NetCDF input files that need to be regridded. For GRIB1 files, it has also been tested for compatibility with the copygb regridding utility mentioned in Section 2.7.

5.2.1 regrid_data_plane usage

The usage statement for the regrid_data_plane utility is shown below:

```
Usage: regrid_data_plane
       input_filename
       to_grid
       output_filename
       -field string
       [-method type]
       [-width n]
       [-gaussian_dx n]
       [-gaussian_radius n]
       [-shape type]
       [-vld_thresh n]
       [-name list]
       [-log file]
       [-v level]
       [-compress level]
```

Required arguments for regrid_data_plane

1. The **input_filename** is the gridded data file to be read.
2. The **to_grid** defines the output grid as a named grid, the path to a gridded data file, or an explicit grid specification string.
3. The **output_filename** is the output NetCDF file to be written.
4. The **-field string** may be used multiple times to define the field(s) to be regridded.

Optional arguments for regrid_data_plane

5. The **-method type** option overrides the default regridding method. Default is NEAREST.
6. The **-width n** option overrides the default regridding width. Default is 1. In case of MAXGAUSS method, the width should be the ratio between from_grid and to_grid (for example, 27 if from_grid is 3km and to_grid is 81.271km).
7. The **-gaussian_dx** option overrides the default delta distance for Gaussian smoothing. Default is 81.271. Ignored if not the MAXGAUSS method.
8. The **-gaussian_radius** option overrides the default radius of influence for Gaussian interpolation. Default is 120. Ignored if not the MAXGAUSS method.

9. The **-shape** option overrides the default interpolation shape. Default is SQUARE.

10. The **-vld_thresh n** option overrides the default required ratio of valid data for regridding. Default is 0.5.

11. The **-name list** specifies a comma-separated list of output variable names for each field specified.

12. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

13. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

14. The **-compress level** option specifies the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

For more details on setting the **to_grid**, **-method**, **-width**, and **-vld_thresh** options, see the **regrid** entry in Section 3.5.1. An example of the regrid_data_plane calling sequence is shown below:

```
regrid_data_plane \
input.grb \
togrid.grb \
regridded.nc \
-field 'name="APCP"; level="A6";'
-field 'name="TMP";  level="Z2";' \
-field 'name="UGRD"; level="Z10";' \
-field 'name="VGRD"; level="Z10";' \
-field 'name="HGT";  level="P500";' \
-method BILIN -width 2 -v 1
```

In this example, the regrid_data_plane tool will regrid data from the **input.grb** file to the grid on which the first record of the **togrid.grb** file resides using Bilinear Interpolation with a width of 2 and write the output in NetCDF format to a file named **regridded.nc**. The variables in **regridded.nc** will include 6-hour accumulated precipitation, 2m temperature, 10m U and V components of the wind, and the 500mb geopotential height.

5.2.2 Automated regridding within tools

While the regrid_data_plane tool is useful as a stand-alone tool, the capability is also included to automatically regrid one or both fields in most of the MET tools that handle gridded data. See the **regrid** entry in Section 3.5 for a description of the configuration file entries that control automated regridding.

5.3 Shift_data_plane tool

The Shift-Data-Plane tool performs a rigid shift of the entire grid based on user-defined specifications and write the field(s) out in NetCDF format. This tool was originally designed to account for track error when comparing fields associated with tropical cyclones. The user specifies the latitude and longitude of the source and destination points to define the shift. Both points must fall within the domain and are used to define the X and Y direction grid unit shift. The shift is then applied to all grid points. The user may specify the method of interpolation and the field to be shifted. The effects of topography and land/water masks are ignored.

5.3.1 shift_data_plane usage

The usage statement for the shift_data_plane utility is shown below:

```
Usage: shift_data_plane
       input_filename
       output_filename
       field_string
       -from lat lon
       -to lat lon
       [-method type]
       [-width n]
       [-log file]
       [-v level]
       [-compress level]
```

shift_data_plane has five required arguments and can also take optional ones.

Required arguments for shift_data_plane

1. The **input_filename** is the gridded data file to be read.
2. The **output_filename** is the output NetCDF file to be written.
3. The **field_string** defines the data to be shifted from the input file.
4. The **-from lat lon** specifies the starting location within the domain to define the shift. Latitude and longitude are defined in degrees North and East, respectively.
5. The **-to lat lon** specifies the ending location within the domain to define the shift. Lat is deg N, Lon is deg E.

Optional arguments for shift_data_plane

6. The **-method type** overrides the default regridding method. Default is NEAREST.

7. The **-width n** overrides the default regridding width. Default is 1.

8. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

9. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

10. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

For more details on setting the **-method** and **-width** options, see the **regrid** entry in Section 3.5.1. An example of the shift_data_plane calling sequence is shown below:

```
shift_data_plane \
nam.grib \
nam_shift_APCP_12.nc \
'name = "APCP"; level = "A12";' \
-from 38.6272  -90.1978 \
-to   40.1717 -105.1092 \
-v 2
```

In this example, the shift_data_plane tool reads 12-hour accumulated precipitation from the **nam.grb** file, applies a rigid shift defined by (38.6272, -90.1978) to (40.1717, -105.1092) and writes the output in NetCDF format to a file named **nam_shift_APCP_12.nc**. These **-from** and **-to** locations result in a grid shift of -108.30 units in the x-direction and 16.67 units in the y-direction.

5.4 MODIS regrid tool

This section contains a description of running the MODIS regrid tool. This tool may be run to create a NetCDF file for use in other MET tools from MODIS level 2 cloud product from NASA. The data browser for these files is: `http://ladsweb.nascom.nasa.gov/`.

5.4.1 modis_regrid usage

The usage statement for the modis_regrid utility is shown below:

```
Usage: modis_regrid
       -data_file path
       -field name
       -out path
       -scale value
       -offset value
       -fill value
       [-units text]
       [-compress level]
       modis_file
```

modis_regrid has some required arguments and can also take optional ones.

Required arguments for modis_regrid

1. The **-data_file path** argument specifies the data files used to get the grid information.
2. The **-field name** argument specifies the name of the field to use in the MODIS data file.
3. The **-out path** argument specifies the name of the output NetCDF file.
4. The **-scale value** argument specifies the scale factor to be used on the raw MODIS values.
5. The **-offset value** argument specifies the offset value to be used on the raw MODIS values.
6. The **-fill value** argument specifies the bad data value in the MODIS data.
7. The **modis_file** argument is the name of the MODIS input file.

Optional arguments for modis_regrid

8. The **-units text** option specifies the units string in the global attributes section of the output file.
9. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

An example of the modis_regrid calling sequence is shown below:

```
modis_regrid -field Cloud_Fraction \
-data_file grid_file \
-out t2.nc \
-units percent \
-scale 0.01 \
-offset 0 \
-fill 127 \
modisfile
```

In this example, the modis_regrid tool will process the Cloud_Fraction field from modisfile and write it out to the output NetCDF file t2.nc on the grid specified in grid_file using the appropriate scale, offset and fill values.

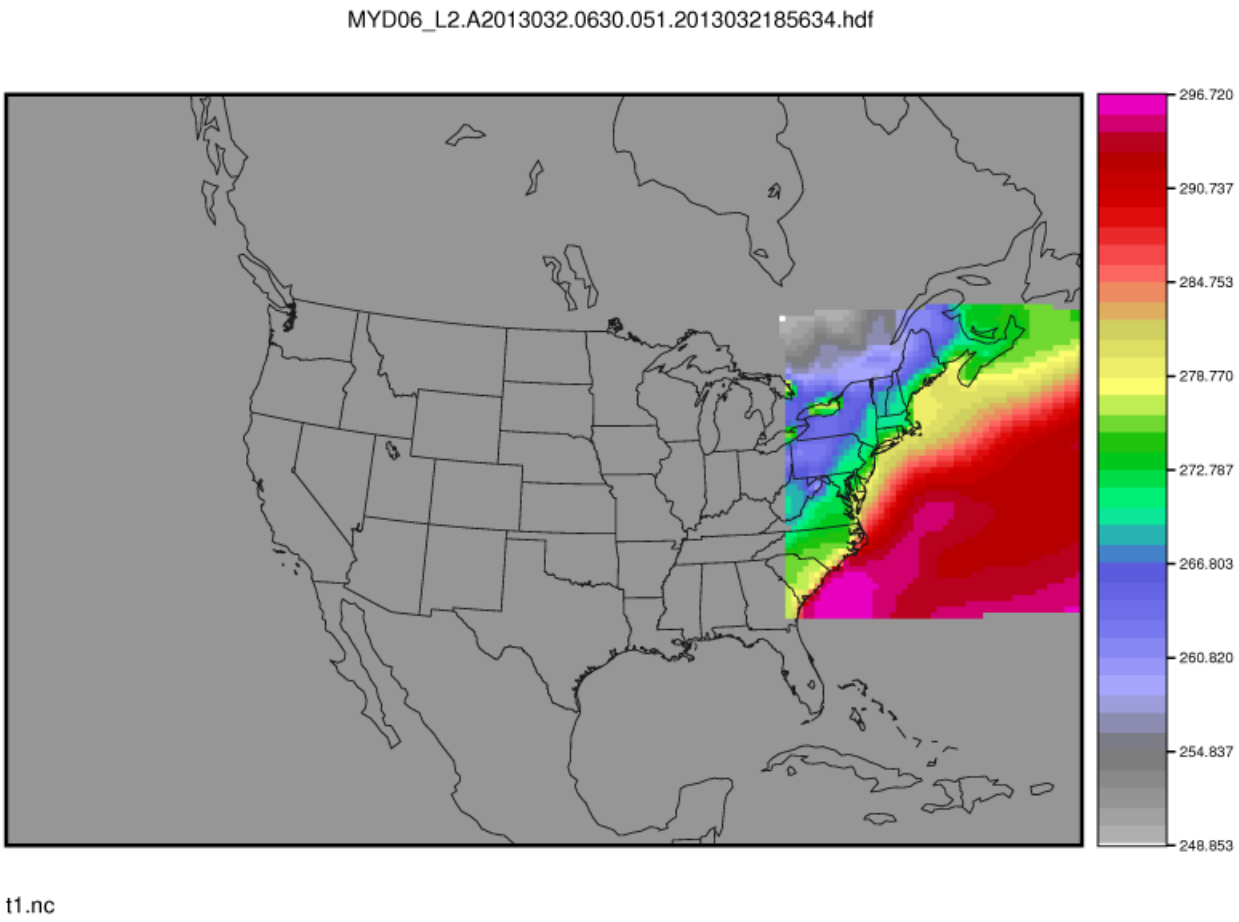

Figure 5.1: Example plot showing surface temperature from a MODIS file.

5.5 WWMCA Tool Documentation

There are two WWMCA tools available. The WWMCA-Plot tool makes a PostScript plot of one or more WWMCA cloud percent files and the WWMCA-Regrid tool regrids binary WWMCA data files and reformats them into NetCDF files that the other MET tools can read. The WWMCA-Regrid tool has been generalized to more broadly support any data stored in the WWMCA binary format.

The WWMCA tools attempt to parse timing and hemisphere information from the file names. They tokenize the filename using underscores (_) and dots (.) and examine each element which need be in no particular order. A string of 10 or more numbers is interpreted as the valid time in YYYYMMDDHH[MMSS] format. The string NH indicates the northern hemisphere while SH indicates the southern hemisphere. While WWMCA data is an analysis and has no forecast lead time, other datasets following this format may. Therefore, a string of 1 to 4 numbers is interpreted as the forecast lead time in hours. While parsing the filename provides default values for this timing information, they can be overridden by explicitly setting their values in the WWMCA-Regrid configuration file.

5.5.1 wwmca_plot usage

The usage statement for the WWMCA-Plot tool is shown below:

```
Usage: wwmca_plot
       [-outdir path]
       [-max max_minutes]
       [-log file]
       [-v level]
       wwmca_cloud_pct_file_list
```

wmmca_plot has some required arguments and can also take optional ones.

Required arguments for wwmca_plot

1. The **wwmca_cloud_pct_file_list** argument represents one or more WWMCA cloud percent files given on the command line. As with any command given to a UNIX shell, the user can use meta-characters as a shorthand way to specify many filenames. For each input file specified, one output PostScript plot will be created.

Optional arguments for wwmca_plot

2. The **-outdir path** option specifies the directory where the output PostScript plots will be placed. If not specified, then the plots will be put in the current (working) directory.

3. The **-max minutes** option specifies the maximum pixel age in minutes to be plotted.

4. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

5. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity will increase the amount of logging.

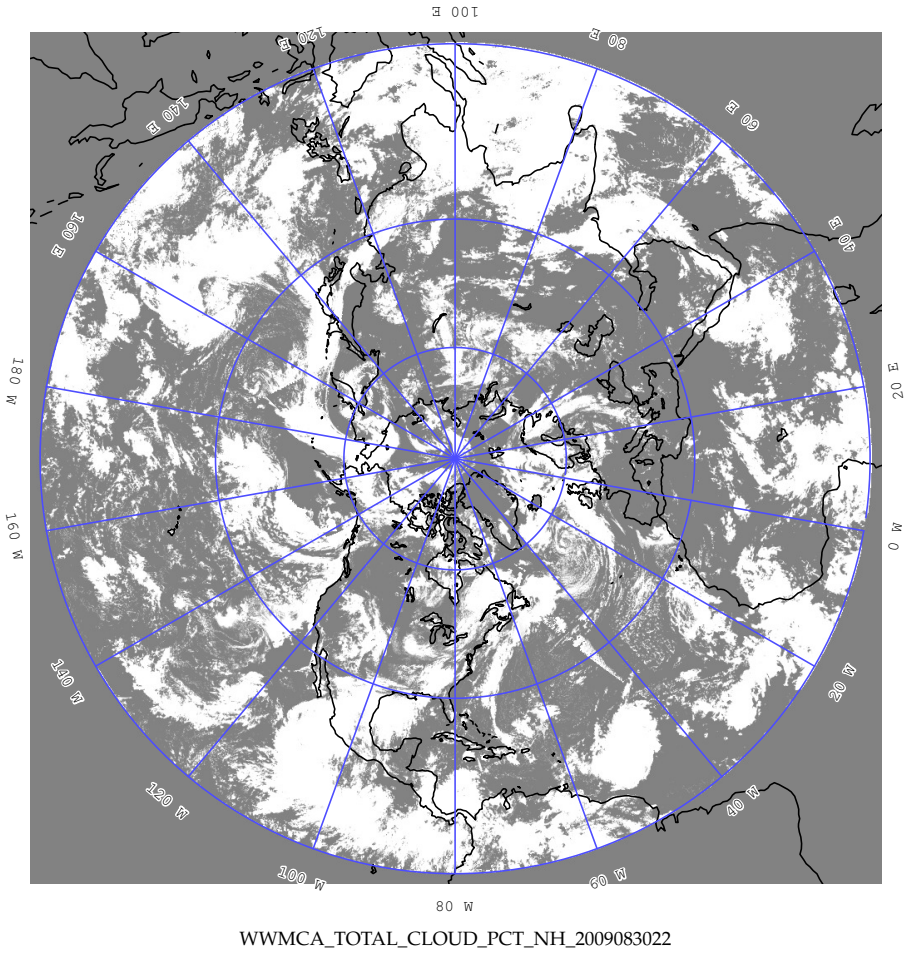

Figure 5.2: Example output of `wwmca_plot` tool.

5.5.2 wwmca_regrid usage

The usage statement for the WWMCA-Regrid tool is shown below:

```
Usage: wwmca_regrid
       -out filename
```

```
-config filename
-nh filename [pt_filename]
-sh filename [pt_filename]
[-log file]
[-v level]
[-compress level]
```

wmmca_regrid has some required arguments and can also take optional ones.

Required arguments for wwmca_regrid

1. The **-out filename** argument specifies the name of the output netCDF file.
2. The **-config filename** argument indicates the name of the configuration file to be used. The contents of the configuration file are discussed below.
3. The **-nh filename [pt_filename]** argument specifies the northern hemisphere WWMCA binary file and, optionally, may be followed by a binary pixel age file. This switch is required if the output grid includes any portion of the northern hemisphere.
4. The **-sh filename [pt_filename]** argument specifies the southern hemisphere WWMCA binary file and, optionally, may be followed by a binary pixel age file. This switch is required if the output grid includes any portion of the southern hemisphere.

Optional arguments for wwmca_regrid

5. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.
6. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity will increase the amount of logging.
7. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

In any regridding problem, there are two grids involved: the "From" grid, which is the grid the input data are on, and the "To" grid, which is the grid the data are to be moved onto. In **WWMCA-Regrid** the "From" grid is pre-defined by the hemisphere of the WWMCA binary files being processed. The "To" grid and corresponding regridding logic are specified using the **regrid** section of the configuration file. If the "To" grid is entirely confined to one hemisphere, then only the WWMCA data file for that hemisphere need be given. If the "To" grid or the interpolation box used straddles the equator the data files for both hemispheres need be given. Once the "To" grid is specified in the config file, the WWMCA-Regrid tool will know which input data files it needs and will complain if it is not given the right ones.

5.5.3 wwmca_regrid configuration file

The default configuration file for the WWMCA-Regrid tool named **WWMCARegridConfig_default** can be found in the installed **share/met/config** directory. We encourage users to make a copy of this file prior to modifying its contents. The contents of the configuration file are described in the subsections below.

Note that environment variables may be used when editing configuration files, as described in Section 4.1.2 for the PB2NC tool.

```
regrid = { ... }
```

See the **regrid** entry in Section 3.5 for a description of the configuration file entries that control regridding.

```
variable_name = "Cloud_Pct";
units         = "percent";
long_name     = "cloud cover percent";
level         = "SFC";
```

The settings listed above are strings which control the output netCDF variable name and specify attributes for that variable.

```
init_time  = "";
valid_time = "";
accum_time = "01";
```

The settings listed above are strings which specify the timing information for the data being processed. The accumulation time is specified in HH[MMSS] format and, by default, is set to a value of 1 hour. The initialization and valid time strings are specified in YYYYMMDD[_HH[MMSS]] format. However, by default they are set to empty strings. If empty, the timing information parsed from the filename will be used. If not empty, these values override the times parsed from the filename.

```
max_minutes     = 120;
swap_endian     = TRUE;
write_pixel_age = FALSE;
```

The settings listed above are control the processing of the WWMCA pixel age data. This data is stored in binary data files in 4-byte blocks. The **swap_endian** option indicates whether the endian-ness of the data should be swapped after reading. The **max_minutes** option specifies a maximum allowed age for the cloud data in minutes. Any data values older than this value are set to bad data in the output. The **write_pixel_age** option writes the pixel age data, in minutes, to the output file instead of the cloud data.

Chapter 6

Regional Verification using Spatial Masking

Verification over a particular region or area of interest may be performed using "masking". Defining a masking region is simply selecting the desired set of grid points to be used. The Gen-Vx-Mask tool automates this process and replaces the Gen-Poly-Mask and Gen-Circle-Mask tools from previous releases. It may be run to create a bitmap verification masking region to be used by many of the statistical tools. This tool enables the user to generate a masking region once for a domain and apply it to many cases. It has been enhanced to support additional types of masking region definition (e.g. tropical-cyclone track over water only). An iterative approach may be used to define complex areas by combining multiple masking regions together.

6.1 Gen-Vx-Mask tool

The Gen-Vx-Mask tool may be run to create a bitmap verification masking region to be used by the the MET statistics tools. This tool enables the user to generate a masking region once for a domain and apply it to many cases. While the MET statistics tools can define some masking regions on the fly use polylines, doing so can be slow, especially for complex polylines containing hundreds of vertices. Using the Gen-Vx-Mask tool to create a bitmap masking region before running the other MET tools will make them run more efficiently.

6.1.1 gen_vx_mask usage

The usage statement for the Gen-Vx-Mask tool is shown below:

```
Usage: gen_vx_mask
       input_file
```

```
mask_file
out_file
[-type str]
[-input_field string]
[-mask_field string]
[-complement]
[-union | -intersection | -symdiff]
[-thresh string]
[-height n]
[-width n]
[-shapeno n]
[-value n]
[-name string]
[-log file]
[-v level]
[-compress level]
```

gen_vx_mask has three required arguments and can take optional ones.

Required arguments for gen_vx_mask

1. The **input_file** argument is a gridded data file which specifies the grid definition for the domain over which the masking bitmap is to be defined. If output from gen_vx_mask, automatically read mask data as the **input_field**.

2. The **mask_file** argument defines the masking information, see below.

   - For "poly", "box", "circle", and "track" masking, specify an ASCII Lat/Lon file.
   - For "grid" and "data" masking, specify a gridded data file.
   - For "solar_alt" and "solar_azi" masking, specify a gridded data file or a time string in YYYYMMDD[_HH[MMSS]] format.
   - For "lat" and "lon" masking, no "mask_file" needed, simply repeat the path for "input_file".
   - For "shape" masking, specify an ESRI shapefile (.shp).

3. The **out_file** argument is the output NetCDF mask file to be written.

Optional arguments for gen_vx_mask

4. The **-type string** option can be used to override the default masking type (poly). See description of supported types below.

5. The **-input_field string** option can be used to read existing mask data from "input_file".

6. The **-mask_field string** option can be used to define the field from "mask_file" to be used for "data" masking.

7. The **-complement** option can be used to to compute the complement of the area defined by "mask_file".

8. The **-union** | **-intersection** | **-symdiff** option can be used to specify how to combine the masks from "input_file" and "mask_file".

9. The **-thresh string** option can be used to define the threshold to be applied.

    - For "circle" and "track" masking, threshold the distance (km).
    - For "data" masking, threshold the values of "mask_field".
    - For "solar_alt" and "solar_azi" masking, threshold the computed solar values.
    - For "lat" and "lon" masking, threshold the latitude and longitude values.

10. The **-height n** and **-width n** options set the size in grid units for "box"masking.

11. The **-shapeno n** option is only used for shapefile masking. (See description of shapefile masking below).

12. The **-value n** option can be used to override the default output mask data value (1).

13. The **-name string** option can be used to specify the output variable name for the mask.

14. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

15. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity will increase the amount of logging.

16. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

The Gen-Vx-Mask tool supports the following types of masking region definition selected using the **-type** command line option:

1. Polyline (**poly**) masking reads an input ASCII file containing Lat/Lon locations, connects the first and last points, and selects grid points falling inside that polyline. This option is useful when defining geographic sub-regions of a domain.

2. Box (**box**) masking reads an input ASCII file containing Lat/Lon locations and draws a box around each point. The height and width of the box is specified by the **-height** and **-width** command line options in grid units. For a square, only one of **-height** or **-width** needs to be used.

3. Circle (**circle**) masking reads an input ASCII file containing Lat/Lon locations and for each grid point, computes the minimum great-circle arc distance in kilometers to those points. If the **-thresh** command line option is not used, the minimum distance value for each grid point will be written to the output. If it is used, only those grid points whose minimum distance meets the threshold criteria will be selected. This option is useful when defining areas within a certain radius of radar locations.

4. Track (**track**) masking reads an input ASCII file containing Lat/Lon locations and for each grid point, computes the minimum great-circle arc distance in kilometers to the track defined by those points. The first and last track points are not connected. As with **circle** masking the output for each grid points depends on the use of the **-thresh** command line option. This option is useful when defining the area within a certain distance of a hurricane track.

5. Grid (**grid**) masking reads an input gridded data file, extracts the field specified using the its grid definition, and selects grid points falling inside that grid. This option is useful when using a model nest to define the corresponding area of the parent domain.

6. Data (**data**) masking reads an input gridded data file, extracts the field specified using the **-mask_field** command line option, thresholds the data using the **-thresh** command line option, and selects grid points which meet that threshold criteria. The option is useful when thresholding topography to define a mask based on elevation or when threshold land use to extract a particular category.

7. Solar altitude (**solar_alt**) and solar azimuth (**solar_azi**) masking computes the solar altitude and azimuth values at each grid point for the time defined by the **mask_file** setting. **mask_file** may either to set to an explicit time string in YYYYMMDD[_HH[MMSS]] format or to a gridded data file. If set to a gridded data file, the **-mask_field** command line option specifies the field of data whose valid time should be used. If the **-thresh** command line option is not used, the raw solar altitude or azimuth value for each grid point will be written to the output. If it is used, the resulting binary mask field will be written. This option is useful when defining a day/night mask.

8. Latitude (**lat**) and longitude (**lon**) masking computes the latitude and longitude value at each grid point. This logic only requires the definition of the grid, specified by the **input_file**. Technically, the **mask_file** is not needed, but a value must be specified for the command line to parse correctly. Users are advised to simple repeat the **input_file** setting twice. If the **-thresh** command line option is not used, the raw latitude or longitude values for each grid point will be written to the output. This option is useful when defining latitude or longitude bands over which to compute statistics.

9. Shapefile (**shape**) masking uses a closed polygon taken from an ESRI shapefile to define the masking region. Gen-Vx-Mask reads the shapefile with the ".shp" suffix and extracts the latitude and longitudes of the vertices. The other types of shapefiles (index file, suffix ".shx", and dBASE file, suffix ".dbf") are not currently used. The shapefile must consist of closed polygons rather than polylines, points, or any of the other data types that shapefiles support. Shapefiles usually contain more than one polygon, and the **-shape n** command line option enables the user to select one polygon from the shapefile. The integer **n** tells which shape number to use from the shapefile. Note that this value is zero-based, so that the first polygon in the shapefile is polygon number 0, the second polygon in the shapefile is polygon number 1, *etc.* For the user's convenience, some utilities that perform human-readable screen dumps of shapefile contents are provided. The **gis_dump_shp**, **gis_dump_shx** and **gis_dump_dbf**

tools enable the user to examine the contents of her shapefiles. As an example, if the user knows the name of the particular polygon he wishes to use but not the number of the polygon in the shapefile, he can use the gis_dump_dbf utility to examine the names of the polygons in the shapefile, and the information written to the screen will tell him what the corresponding polygon number is.

The polyline, box, circle, and track masking methods all read an ASCII file containing Lat/Lon locations. Those files must contain a string, which defines the name of the masking region, followed by a series of whitespace-separated latitude (degrees north) and longitude (degree east) values.

The Gen-Vx-Mask tool performs three main steps, described below.

1. Determine the **input_field** and grid definition.
   - Read the **input_file** to determine the grid over which the mask should be defined.
   - By default, initialize the **input_field** at each grid point to a value of zero.
   - If the **-input_field** option was specified, initialize the **input_field** at each grid point to the value of that field.
   - If the **input_file** is the output from a previous run of Gen-Vx-Mask, automatically initialize each grid point with the **input_field** value.
2. Determine the **mask_field**.
   - Read the **mask_file**, process it based on the **-type** setting (as described above), and define the **mask_field** value for each grid point to specify whether or not it is included in the mask.
   - By default, store the mask value as 1 unless the **-value** option was specified to override that default value.
   - If the **-complement** option was specified, the opposite of the masking area is selected.
3. Apply logic to combine the **input_field** and **mask_field** and write the **out_file**.
   - By default, the output value at each grid point is set to the value of **mask_field** if included in the mask, or the value of **input_field** if not included.
   - If the **-union**, **-intersection**, or **-symdiff** option was specified, apply that logic to the **input_field** and **mask_field** values at each grid point to determine the output value.
   - Write the output value for each grid point to the **out_file**.

This three step process enables the Gen-Vx-Mask tool to be run iteratively on its own output to generate complex masking areas. Additionally, the **-union**, **-intersection**, and **-symdiff** options control the logic for combining the input data value and current mask value at each grid point. For example, one could define a complex masking region by selecting grid points with an elevation greater than 1000 meters within a specified geographic region by doing the following:

- Run the Gen-Vx-Mask tool to apply data masking by thresholding a field of topography greater than 1000 meters.

- Rerun the Gen-Vx-Mask tool passing in the output of the first call and applying polyline masking to define the geographic area of interest.
  - Use the **-intersection** option to only select grid points whose value is non-zero in both the input field and the current mask.

An example of the gen_vx_mask calling sequence is shown below:

```
gen_vx_mask sample_fcst.grb \
CONUS.poly CONUS_poly.nc
```

In this example, the Gen-Vx-Mask tool will read the ASCII Lat/Lon file named **CONUS.poly** and apply the default polyline masking method to the domain on which the data in the file **sample_fcst.grib** resides. It will create a NetCDF file containing a bitmap for the domain with a value of 1 for all grid points inside the CONUS polyline and a value of 0 for all grid points outside. It will write an output NetCDF file named **CONUS_poly.nc.**

6.2 Feature-Relative Methods

This section contains a description of several methods that may be used to perform feature-relative (or event -based) evaluation. The methodology pertains to examining the environment surrounding a particular feature or event such as a tropical, extra-tropical cyclone, convective cell, snow-band, etc. Several approaches are available for these types of investigations including applying masking described above (e.g. circle or box) or using the "FORCE" interpolation method in the regrid configuration option (see 3.5.1). These methods generally require additional scripting, including potentially storm-track identification, outside of MET to be paired with the features of the MET tools.

Chapter 7

Point-Stat Tool

7.1 Introduction

The Point-Stat tool provides verification statistics for forecasts at observation points (as opposed to over gridded analyses). The Point-Stat tool matches gridded forecasts to point observation locations and supports several different interpolation options. The tool then computes continuous, categorical, spatial, and probabilistic verification statistics. The categorical and probabilistic statistics generally are derived by applying a threshold to the forecast and observation values. Confidence intervals - representing the uncertainty in the verification measures - are computed for the verification statistics.

Scientific and statistical aspects of the Point-Stat tool are discussed in the following section. Practical aspects of the Point-Stat tool are described in Section7.3.

7.2 Scientific and statistical aspects

The statistical methods and measures computed by the Point-Stat tool are described briefly in this section. In addition, Section 7.2.1 discusses the various interpolation options available for matching the forecast grid point values to the observation points. The statistical measures computed by the Point-Stat tool are described briefly in Section 7.2.3 and in more detail in Appendix C. Section 7.2.4 describes the methods for computing confidence intervals that are applied to some of the measures computed by the Point-Stat tool; more detail on confidence intervals is provided in Appendix D.

7.2.1 Interpolation/matching methods

This section provides information about the various methods available in MET to match gridded model output to point observations. Matching in the vertical and horizontal are completed separately using different methods.

In the vertical, if forecasts and observations are at the same vertical level, then they are paired as-is. If any discrepancy exists between the vertical levels, then the forecasts are interpolated to the level of the observation. The vertical interpolation is done in natural log of pressure coordinates, except for specific humidity, which is interpolated using the natural log of specific humidity in natural log of pressure coordinates. Vertical interpolation for heights above ground are done linear in height coordinates. When forecasts are for the surface, no interpolation is done. They are matched to observations with message types that are mapped to **SURFACE** in the **message_type_group_map** configuration option. By default, the surface message types include ADPSFC, SFCSHP, and MSONET.

To match forecasts and observations in the horizontal plane, the user can select from a number of methods described below. Many of these methods require the user to define the width of the forecast grid W, around each observation point P, that should be considered. In addition, the user can select the interpolation shape, either a SQUARE or a CIRCLE. For example, a square of width 2 defines the 2 x 2 set of grid points enclosing P, or simply the 4 grid points closest to P. A square of width of 3 defines a 3 x 3 square consisting of 9 grid points centered on the grid point closest to P. Figure 7.1 provides illustration. The point P denotes the observation location where the interpolated value is calculated. The interpolation width W, shown is five.

This section describes the options for interpolation in the horizontal.

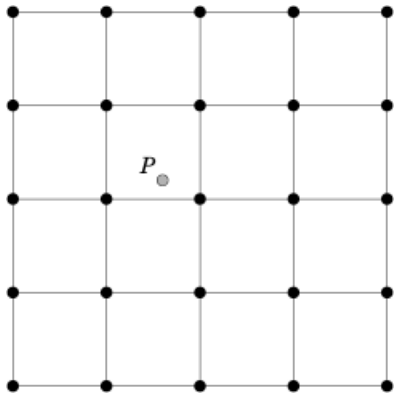

Figure 7.1: Diagram illustrating matching and interpolation methods used in MET. See text for explanation.

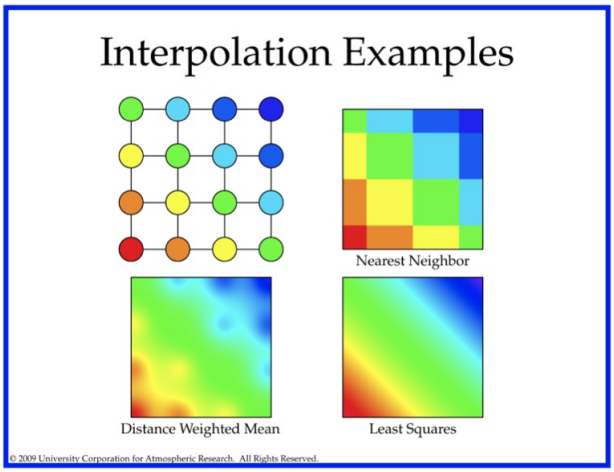

Figure 7.2: Illustration of some matching and interpolation methods used in MET. See text for explanation.

Nearest Neighbor

The forecast value at P is assigned the value at the nearest grid point. No interpolation is performed. Here, "nearest" means spatially closest in horizontal grid coordinates. This method is used by default when the interpolation width, W, is set to 1.

Geography Match

The forecast value at P is assigned the value at the nearest grid point in the interpolation area where the land/sea mask and topography criteria are satisfied.

Gaussian

The forecast value at P is a weighted sum of the values in the interpolation area. The weight given to each forecast point follows the Gaussian distribution with nearby points contributing more the far away points. The shape of the distribution is configured using sigma.

When used for regridding, with the **regrid** configuration option, or smoothing, with the **interp** configuration option in grid-to-grid comparisons, the Gaussian method is named **MAXGAUSS** and is implemented as a 2-step process. First, the data is regridded or smoothed using the maximum value interpolation method described below, where the **width** and **shape** define the interpolation area. Second, the Gaussian smoother, defined by the **gaussian_dx** and **gaussian_radius** configuration options, is applied.

Minimum value

The forecast value at P is the minimum of the values in the interpolation area.

Maximum value

The forecast value at P is the maximum of the values in the interpolation area.

Distance-weighted mean

The forecast value at P is a weighted sum of the values in the interpolation area. The weight given to each forecast point is the reciprocal of the square of the distance (in grid coordinates) from P. The weighted sum of forecast values is normalized by dividing by the sum of the weights.

Unweighted mean

This method is similar to the distance-weighted mean, except all the weights are equal to 1. The distance of any point from P is not considered.

Median

The forecast value at P is the median of the forecast values in the interpolation area.

Least-Squares Fit

To perform least squares interpolation of a gridded field at a location P, MET uses an **WxW** subgrid centered (as closely as possible) at P. Figure 7.1 shows the case where N = 5.

If we denote the horizontal coordinate in this subgrid by x, and vertical coordinate by y, then we can assign coordinates to the point P relative to this subgrid. These coordinates are chosen so that the center of the grid is. For example, in Figure 7.2, P has coordinates (-0.4, 0.2). Since the grid is centered near P, the coordinates of P should always be at most 0.5 in absolute value. At each of the vertices of the grid (indicated by black dots in the figure), we have data values. We would like to use these values to interpolate a value at P. We do this using least squares. If we denote the interpolated value by z, then we fit an expression of the form $z = \alpha(x) + \beta(y) + \gamma$ over the subgrid. The values of α, β, γ are calculated from the data values at the vertices. Finally, the coordinates (**x,y**) of P are substituted into this expression to give z, our least squares interpolated data value at P.

Bilinear Interpolation

This method is performed using the four closest grid squares. The forecast values are interpolated linearly first in one dimension and then the other to the location of the observation.

Upper Left, Upper Right, Lower Left, Lower Right Interpolation

This method is performed using the four closest grid squares. The forecast values are interpolated to the specified grid point.

Best Interpolation

The forecast value at P is the chosen as the grid point inside the interpolation area whose value most closely matches the observation value.

7.2.2 HiRA framework

The Point-Stat tool has been enhanced to include the High Resolution Assessment (HiRA) verification logic (Mittermaier, 2014). HiRA is analogous to neighborhood verification but for point observations. The HiRA logic interprets the forecast values surrounding each point observation as an ensemble forecast. These ensemble values are processed in two ways. First, the ensemble continuous statistics (ECNT) and the ranked probability score (RPS) line types are computed directly from the ensemble values. Second, for each categorical threshold specified, a fractional coverage value is computed as the ratio of the nearby forecast values that meet the threshold criteria. Point-Stat evaluates those fractional coverage values as if they were a probability forecast. When applying HiRA, users should enable the matched pair (MPR), probabilistic (PCT, PSTD, PJC, or PRC), continuous ensemble statistics (ECNT), or ranked probability score (RPS) line types in the output_flag dictionary. The number of probabilistic HiRA output lines is determined by the number of categorical forecast thresholds and HiRA neighborhood widths chosen.

The HiRA framework provides a unique method for evaluating models in the neighborhood of point observations, allowing for some spatial and temporal uncertainty in the forecast and/or the observations. Additionally, the HiRA framework can be used to compare deterministic forecasts to ensemble forecasts. In MET, the neighborhood is a circle or square centered on the grid point closest to the observation location. An event is defined, then the proportion of points with events in the neighborhood is calculated. This proportion is treated as an ensemble probability, though it is likely to be uncalibrated.

Figure1.3 shows a couple of examples of how the HiRA proportion is derived at a single model level using square neighborhoods. Events (in our case, model accretion values > 0) are separated from non-events (model accretion value $= 0$). Then, in each neighborhood, the total proportion of events is calculated. In the

leftmost panel, four events exist in the 25 point neighborhood, making the HiRA proportion is $4/25 = 0.16$. For the neighborhood of size 9 centered in that same panel, the HiRA proportion is $1/9$. In the right panel, the size 25 neighborhood has HiRA proportion of $6/25$, with the centered 9-point neighborhood having a HiRA value of $2/9$. To extend this method into 3-dimensions, all layers within the user-defined layer are also included in the calculation of the proportion in the same manner.

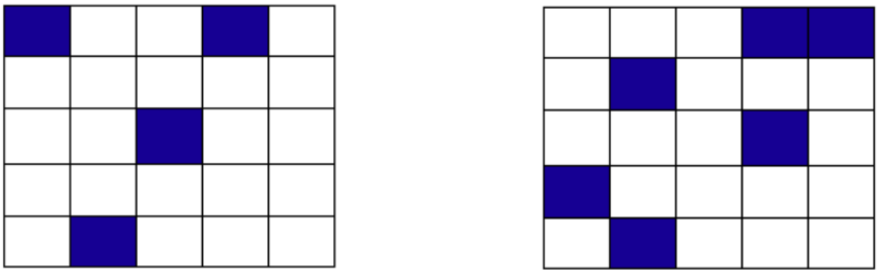

Figure 7.3: Example showing how HiRA proportions are calculated.

Often, the neighborhood size is chosen so that multiple models to be compared have approximately the same horizontal resolution. Then, standard metrics for probabilistic forecasts, such as Brier Score, can be used to compare those forecasts. HiRA was developed using surface observation stations so the neighborhood lies completely within the horizontal plane. With any type of upper air observation, the vertical neighborhood must also be defined.

7.2.3 Statistical measures

The Point-Stat tool computes a wide variety of verification statistics. Broadly speaking, these statistics can be subdivided into statistics for categorical variables and statistics for continuous variables. The categories of measures are briefly described here; specific descriptions of the measures are provided in Appendix C. Additional information can be found in Wilks (2011) and Jolliffe and Stephenson (2003), and on the world-wide web at

`http://www.bom.gov.au/bmrc/wefor/staff/eee/verif/verif_web_page.html`.

In addition to these verification measures, the Point-Stat tool also computes partial sums and other FHO statistics that are produced by the NCEP verification system. These statistics are also described in Appendix C.

Measures for categorical variables

Categorical verification statistics are used to evaluate forecasts that are in the form of a discrete set of categories rather than on a continuous scale. If the original forecast is continuous, the user may specify one or more threhsolds in the configuration file to divide the continuous measure into categories. Currently, Point-Stat computes categorical statistics for variables in two or more categories. The special case of dichotomous

(i.e., 2-category) variables has several types of statistics calculated from the resulting contingency table and are available in the CTS output line type. For multi-category variables, fewer statistics can be calculated so these are available separately, in line type MCTS. Categorical variables can be intrinsic (e.g., rain/no-rain) or they may be formed by applying one or more thresholds to a continuous variable (e.g., temperature < 273.15 K or cloud coverage percentages in 10% bins). See Appendix C for more information.

Measures for continuous variables

For continuous variables, many verification measures are based on the forecast error (i.e., f - o). However, it also is of interest to investigate characteristics of the forecasts, and the observations, as well as their relationship. These concepts are consistent with the general framework for verification outlined by Murphy and Winkler (1987). The statistics produced by MET for continuous forecasts represent this philosophy of verification, which focuses on a variety of aspects of performance rather than a single measure. See Appendix C for specific information.

A user may wish to eliminate certain values of the forecasts from the calculation of statistics, a process referred to here as"'conditional verification". For example, a user may eliminate all temperatures above freezing and then calculate the error statistics only for those forecasts of below freezing temperatures. Another common example involves verification of wind forecasts. Since wind direction is indeterminate at very low wind speeds, the user may wish to set a minimum wind speed threshold prior to calculating error statistics for wind direction. The user may specify these threhsolds in the configuration file to specify the conditional verification. Thresholds can be specified using the usual Fortran conventions (<, <=, ==, !-, >=, or >) followed by a numeric value. The threshold type may also be specified using two letter abbreviations (lt, le, eq, ne, ge, gt). Further, more complex thresholds can be achieved by defining multiple thresholds and using && or || to string together event definition logic. The forecast and observation threshold can be used together according to user preference by specifying one of: UNION, INTERSECTION, or SYMDIFF (symmetric difference).

Measures for probabilistic forecasts and dichotomous outcomes

For probabilistic forecasts, many verification measures are based on reliability, accuracy and bias. However, it also is of interest to investigate joint and conditional distributions of the forecasts and the observations, as in Wilks (2011). See Appendix C for specific information.

Probabilistic forecast values are assumed to have a range of either 0 to 1 or 0 to 100. If the max data value is > 1, we assume the data range is 0 to 100, and divide all the values by 100. If the max data value is <= 1, then we use the values as is. Further, thresholds are applied to the probabilities with equality on the lower end. For example, with a forecast probability p, and thresholds t1 and t2, the range is defined as: t1 <= p < t2. The exception is for the highest set of thresholds, when the range includes 1: t1 <= p <= 1. To make configuration easier, in METv6.0, these probabilities may be specified in the configuration file as a list (>0.00,>0.25,>0.50,>0.75,>1.00) or using shorthand notation (==0.25) for bins of equal width.

When the "prob" entry is set as a dictionary to define the field of interest, setting "prob_as_scalar = TRUE" indicates that this data should be processed as regular scalars rather than probabilities. For example, this option can be used to compute traditional 2x2 contingency tables and neighborhood verification statistics for probability data. It can also be used to compare two probability fields directly.

Measures for comparison against climatology

For each of the types of statistics mentioned above (categorical, continuous, and probabilistic), it is possible to calculate measures of skill relative to climatology. MET will accept a climatology file provided by the user, and will evaluate it as a reference forecast. Further, anomalies, i.e. departures from average conditions, can be calculated. As with all other statistics, the available measures will depend on the nature of the forecast. Common statistics that use a climatological reference include: the mean squared error skill score (MSESS), the Anomaly Correlation (ANOM_CORR), scalar and vector anomalies (SAL1L2 and VAL1L2), continuous ranked probability skill score (CRPSS), Brier Skill Score (BSS) (Wilks, 2011; Mason, 2004).

Often, the sample climatology is used as a reference by a skill score. The sample climatology is the average over all included observations and may be transparent to the user. This is the case in most categorical skill scores. The sample climatology will probably prove more difficult to improve upon than a long term climatology, since it will be from the same locations and time periods as the forecasts. This may mask legitimate forecast skill. However, a more general climatology, perhaps covering many years, is often easier to improve upon and is less likely to mask real forecast skill.

7.2.4 Statistical confidence intervals

A single summary score gives an indication of the forecast performance, but it is a single realization from a random process that neglects uncertainty in the score's estimate. That is, it is possible to obtain a good score, but it may be that the "good" score was achieved by chance and does not reflect the "true" score. Therefore, when interpreting results from a verification analysis, it is imperative to analyze the uncertainty in the realized scores. One good way to do this is to utilize confidence intervals. A confidence interval indicates that if the process were repeated many times, say 100, then the true score would fall within the interval $100(1-\alpha)\%$ of the time. Typical values of α are 0.01, 0.05, and 0.10. The Point-Stat tool allows the user to select one or more specific α-values to use.

For continuous fields (e.g., temperature), it is possible to estimate confidence intervals for some measures of forecast performance based on the assumption that the data, or their errors, are normally distributed. The Point-Stat tool computes confidence intervals for the following summary measures: forecast mean and standard deviation, observation mean and standard deviation, correlation, mean error, and the standard deviation of the error. In the case of the respective means, the central limit theorem suggests that the means are normally distributed, and this assumption leads to the usual $100(1-\alpha)\%$ confidence intervals for the mean. For the standard deviations of each field, one must be careful to check that the field of interest is normally distributed, as this assumption is necessary for the interpretation of the resulting confidence intervals.

For the measures relating the two fields (i.e., mean error, correlation and standard deviation of the errors), confidence intervals are based on either the joint distributions of the two fields (e.g., with correlation) or on a function of the two fields. For the correlation, the underlying assumption is that the two fields follow a bivariate normal distribution. In the case of the mean error and the standard deviation of the mean error, the assumption is that the errors are normally distributed, which for continuous variables, is usually a reasonable assumption, even for the standard deviation of the errors.

Bootstrap confidence intervals for any verification statistic are available in MET. Bootstrapping is a non-parametric statistical method for estimating parameters and uncertainty information. The idea is to obtain a sample of the verification statistic(s) of interest (e.g., bias, ETS, etc.) so that inferences can be made from this sample. The assumption is that the original sample of matched forecast-observation pairs is representative of the population. Several replicated samples are taken with replacement from this set of forecast-observation pairs of variables (e.g., precipitation, temperature, etc.), and the statistic(s) are calculated for each replicate. That is, given a set of n forecast-observation pairs, we draw values at random from these pairs, allowing the same pair to be drawn more than once, and the statistic(s) is (are) calculated for each replicated sample. This yields a sample of the statistic(s) based solely on the data without making any assumptions about the underlying distribution of the sample. It should be noted, however, that if the observed sample of matched pairs is dependent, then this dependence should be taken into account somehow. Currently, in the confidence interval methods in MET do not take into account dependence, but future releases will support a robust method allowing for dependence in the original sample. More detailed information about the bootstrap algorithm is found in the appendix.

Confidence intervals can be calculated from the sample of verification statistics obtained through the bootstrap algorithm. The most intuitive method is to simply take the appropriate quantiles of the sample of statistic(s). For example, if one wants a 95% CI, then one would take the 2.5 and 97.5 percentiles of the resulting sample. This method is called the percentile method, and has some nice properties. However, if the original sample is biased and/or has non-constant variance, then it is well known that this interval is too optimistic. The most robust, accurate, and well-behaved way to obtain accurate CIs from bootstrapping is to use the bias corrected and adjusted percentile method (or BCa). If there is no bias, and the variance is constant, then this method will yield the usual percentile interval. The only drawback to the approach is that it is computationally intensive. Therefore, both the percentile and BCa methods are available in MET, with the considerably more efficient percentile method being the default.

The only other option associated with bootstrapping currently available in MET is to obtain replicated samples smaller than the original sample (i.e., to sample $m < n$ points at each replicate). Ordinarily, one should use $m = n$, and this is the default. However, there are cases where it is more appropriate to use a smaller value of m (e.g., when making inference about high percentiles of the original sample). See Gilleland (2008) for more information and references about this topic.

MET provides parametric confidence intervals based on assumptions of normality for the following categorical statistics:

- Base Rate
- Forecast Mean

- Accuracy
- Probability of Detection
- Probability of Detection of the non-event
- Probability of False Detection
- False Alarm Ratio
- Critical Success Index
- Hanssen-Kuipers Discriminant
- Odds Ratio
- Log Odds Ratio
- Odds Ratio Skill Score
- Extreme Dependency Score
- Symmetric Extreme Dependency Score
- Extreme Dependency Index
- Symmetric Extremal Dependency Index

MET provides parametric confidence intervals based on assumptions of normality for the following continuous statistics:

- Forecast and Observation Means
- Forecast, Observation, and Error Standard Deviations
- Pearson Correlation Coefficient
- Mean Error

MET provides parametric confidence intervals based on assumptions of normality for the following probabilistic statistics:

- Brier Score
- Base Rate

MET provides non-parametric bootstrap confidence intervals for many categorical and continuous statistics. Kendall's Tau and Spearman's Rank correlation coefficients are the only exceptions. Computing bootstrap confidence intervals for these statistics would be computationally unrealistic.

For more information on confidence intervals pertaining to verification measures, see Wilks (2011), Jolliffe and Stephenson (2003), and Bradley (2008).

7.3 Practical information

The Point-Stat tool is used to perform verification of a gridded model field using point observations. The gridded model field to be verified must be in one of the supported file formats. The point observations must be formatted as the NetCDF output of the point reformatting tools described in Chapter 4. The Point-Stat tool provides the capability of interpolating the gridded forecast data to the observation points using a variety of methods as described in Section 7.2.1. The Point-Stat tool computes a number of continuous statistics on the matched pair data as well as discrete statistics once the matched pair data have been thresholded.

7.3.1 point_stat usage

The usage statement for the Point-Stat tool is shown below:

```
Usage: point_stat
       fcst_file
       obs_file
       config_file
       [-point_obs file]
       [-obs_valid_beg time]
       [-obs_valid_end time]
       [-outdir path]
       [-log file]
       [-v level]
```

point_stat has three required arguments and can take many optional ones.

Required arguments for point_stat

1. The **fcst_file** argument names the gridded file in either GRIB or NetCDF containing the model data to be verified.

2. The **obs_file** argument indicates the NetCDF file (output of PB2NC or ASCII2NC) containing the point observations to be used for verifying the model.

3. The **config_file** argument indicates the name of the configuration file to be used. The contents of the configuration file are discussed below.

Optional arguments for point_stat

4. The **-point_obs file** may be used to pass additional NetCDF point observation files to be used in the verification.

5. The **-obs_valid_beg time** option in YYYYMMDD[_HH[MMSS]] format sets the beginning of the observation matching time window, overriding the configuration file setting.

6. The **-obs_valid_end time** option in YYYYMMDD[_HH[MMSS]] format sets the end of the observation matching time window, overriding the configuration file setting.

7. The **-outdir path** indicates the directory where output files should be written.

8. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

9. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity will increase the amount of logging.

An example of the point_stat calling sequence is shown below:

```
point_stat sample_fcst.grb \
sample_pb.nc \
PointStatConfig
```

In this example, the Point-Stat tool evaluates the model data in the sample_fcst.grb GRIB file using the observations in the NetCDF output of PB2NC, sample_pb.nc, applying the configuration options specified in the PointStatConfig file.

7.3.2 point_stat configuration file

The default configuration file for the Point-Stat tool named **PointStatConfig_default** can be found in the installed **share/met/config** directory. Another version is located in **scripts/config**. We encourage users to make a copy of these files prior to modifying their contents. The contents of the configuration file are described in the subsections below.

Note that environment variables may be used when editing configuration files, as described in Section 4.1.2 for the PB2NC tool.

```
model          = "WRF";
desc           = "NA";
regrid         = { ... }
climo_mean     = { ... }
climo_stdev    = { ... }
```

```
climo_cdf      = { ... }
obs_window     = { beg = -5400; end =  5400; }
mask           = { grid = [ "FULL" ]; poly = []; sid = []; }
ci_alpha       = [ 0.05 ];
boot           = { interval = PCTILE; rep_prop = 1.0; n_rep = 1000;
                   rng = "mt19937"; seed = ""; }
interp         = { vld_thresh = 1.0; shape = SQUARE;
                   type = [ { method = NEAREST; width = 1; } ]; }
censor_thresh  = [];
censor_val     = [];
eclv_points    = 0.05;
rank_corr_flag = TRUE;
sid_inc        = [];
sid_exc        = [];
duplicate_flag = NONE;
obs_quality    = [];
obs_summary    = NONE;
obs_perc_value = 50;
message_type_group_map = [...];
tmp_dir        = "/tmp";
output_prefix  = "";
version        = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

Setting up the **fcst** and **obs** dictionaries of the configuration file is described in Section 3.5.1. The following are some special consideration for the Point-Stat tool.

The **obs** dictionary looks very similar to the **fcst** dictionary. When the forecast and observation variables follow the same naming convention, one can easily copy over the forecast settings to the observation dictionary using **obs** = **fcst;**. However when verifying forecast data in NetCDF format or verifying against not-standard observation variables, users will need to specify the **fcst** and **obs** dictionaries separately. The number of fields specified in the **fcst** and **obs** dictionaries must match.

The **message_type** entry, defined in the **obs** dictionary, contains a comma-separated list of the message types to use for verification. At least one entry must be provided. The Point-Stat tool performs verification using observations for one message type at a time. See `http://www.emc.ncep.noaa.gov/mmb/data_processing/PrepBUFR.doc/table_1.htm` for a list of the possible types. If using **obs** = **fcst;**, it can be defined in the forecast dictionary and the copied into the observation dictionary.

```
land_mask = {
   flag      = FALSE;
   file_name = [];
   field     = { name = "LAND"; level = "L0"; }
   regrid    = { method = NEAREST; width = 1; }
   thresh = eq1;
}
```

The **land_mask** dictionary defines the land/sea mask field which is used when verifying at the surface. For point observations whose message type appears in the **LANDSF** entry of the **message_type_group_map** setting, only use forecast grid points where land = TRUE. For point observations whose message type appears in the **WATERSF** entry of the **message_type_group_map** setting, only use forecast grid points where land = FALSE. The **flag** entry enables/disables this logic. If the **file_name** is left empty, then the land/sea is assumed to exist in the input forecast file. Otherwise, the specified file(s) are searched for the data specified in the **field** entry. The **regrid** settings specify how this field should be regridded to the verification domain. Lastly, the **thresh** entry is the threshold which defines land (threshold is true) and water (threshold is false).

```
topo_mask = {
   flag               = FALSE;
   file_name          = [];
   field              = { name = "TOPO"; level = "L0"; }
   regrid             = { method = BILIN; width = 2; }
   use_obs_thresh     = ge-100&&le100;
   interp_fcst_thresh = ge-50&&le50;
}
```

The **topo_mask** dictionary defines the model topography field which is used when verifying at the surface. This logic is applied to point observations whose message type appears in the **SURFACE** entry of the **message_type_group_map** setting. Only use point observations where the topo - station elevation difference meets the **use_obs_thresh** threshold entry. For the observations kept, when interpolating forecast data to the observation location, only use forecast grid points where the topo - station difference meets the **interp_fcst_thresh** threshold entry. The **flag** entry enables/disables this logic. If the **file_name** is left empty, then the topography data is assumed to exist in the input forecast file. Otherwise, the specified file(s) are searched for the data specified in the **field** entry. The **regrid** settings specify how this field should be regridded to the verification domain.

```
hira = {
```

```
   flag            = FALSE;
   width           = [ 2, 3, 4, 5 ]
   vld_thresh      = 1.0;
   cov_thresh      = [ ==0.25 ];
   shape           = SQUARE;
   prob_cat_thresh = [];
}
```

The **hira** dictionary that is very similar to the **interp** and **nbrhd** entries. It specifies information for applying the High Resolution Assessment (HiRA) verification logic described in section 7.2.2. The **flag** entry is a boolean which toggles HiRA on (**TRUE**) and off (**FALSE**). The **width** and **shape** entries define the neighborhood size and shape, respectively. Since HiRA applies to point observations, the width may be even or odd. The **vld_thresh** entry is the required ratio of valid data within the neighborhood to compute an output value. The **cov_thresh** entry is an array of probabilistic thresholds used to populate the Nx2 probabilistic contingency table written to the PCT output line and used for computing probabilistic statistics. The **prob_cat_thresh** entry defines the thresholds to be used in computing the ranked probability score in the RPS output line type. If left empty but climatology data is provided, the **climo_cdf** thresholds will be used instead of **prob_cat_thresh**.

```
output_flag = {
   fho    = BOTH;
   ctc    = BOTH;
   cts    = BOTH;
   mctc   = BOTH;
   mcts   = BOTH;
   cnt    = BOTH;
   sl1l2  = BOTH;
   sal1l2 = BOTH;
   vl1l2  = BOTH;
   vcnt   = BOTH;
   val1l2 = BOTH;
   pct    = BOTH;
   pstd   = BOTH;
   pjc    = BOTH;
   prc    = BOTH;
   ecnt   = BOTH;  // Only for HiRA
   rps    = BOTH;  // Only for HiRA
   eclv   = BOTH;
   mpr    = BOTH;
}
```

The **output_flag** array controls the type of output that the Point-Stat tool generates. Each flag corresponds to an output line type in the STAT file. Setting the flag to NONE indicates that the line type should not be generated. Setting the flag to STAT indicates that the line type should be written to the STAT file only. Setting the flag to BOTH indicates that the line type should be written to the STAT file as well as a separate ASCII file where the data is grouped by line type. The output flags correspond to the following output line types:

1. **FHO** for Forecast, Hit, Observation Rates
2. **CTC** for Contingency Table Counts
3. **CTS** for Contingency Table Statistics
4. **MCTC** for Multi-category Contingency Table Counts
5. **MCTS** for Multi-category Contingency Table Statistics
6. **CNT** for Continuous Statistics
7. **SL1L2** for Scalar L1L2 Partial Sums
8. **SAL1L2** for Scalar Anomaly L1L2 Partial Sums when climatological data is supplied
9. **VL1L2** for Vector L1L2 Partial Sums
10. **VCNT** for Vector Continuous Statistics (Note that bootstrap confidence intervals are not currently calculated for this line type.)
11. **VAL1L2** for Vector Anomaly L1L2 Partial Sums when climatological data is supplied
12. **PCT** for Contingency Table counts for Probabilistic forecasts
13. **PSTD** for contingency table Statistics for Probabilistic forecasts with Dichotomous outcomes
14. **PJC** for Joint and Conditional factorization for Probabilistic forecasts
15. **PRC** for Receiver Operating Characteristic for Probabilistic forecasts
16. **ECNT** for Ensemble Continuous Statistics is only computed for the HiRA methodology
17. **RPS** for Ranked Probability Score is only computed for the HiRA methodology
18. **ECLV** for Economic Cost/Loss Relative Value
19. **MPR** for Matched Pair data

Note that the first two line types are easily derived from each other. Users are free to choose which measures are most desired. The output line types are described in more detail in Section 7.3.3.

Note that writing out matched pair data (MPR lines) for a large number of cases is generally not recommended. The MPR lines create very large output files and are only intended for use on a small set of cases.

If all line types corresponding to a particular verification method are set to NONE, the computation of those statistics will be skipped in the code and thus make the Point-Stat tool run more efficiently. For example, if FHO, CTC, and CTS are all set to NONE, the Point-Stat tool will skip the categorical verification step.

7.3.3 point_stat output

point_stat produces output in STAT and, optionally, ASCII format. The ASCII output duplicates the STAT output but has the data organized by line type. The output files will be written to the default output directory or the directory specified using the "-outdir" command line option.

The output STAT file will be named using the following naming convention:

point_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV.stat where PREFIX indicates the user-defined output prefix, HHMMSSL indicates the forecast lead time and YYYYMMDD_HHMMSS indicates the forecast valid time.

The output ASCII files are named similarly:

point_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV_TYPE.txt where TYPE is one of mpr, fho, ctc, cts, cnt, mctc, mcts, pct, pstd, pjc, prc, ecnt, rps, eclv, sl1l2, sal1l2, vl1l2, vcnt or val1l2 to indicate the line type it contains.

The first set of header columns are common to all of the output files generated by the Point-Stat tool. Tables describing the contents of the header columns and the contents of the additional columns for each line type are listed in the following tables. The ECNT line type is described in Section 9.2. The RPS line type is described in Section 9.3.

Table 7.1: Header information for each file point-stat outputs.

HEADER		
Column Number	**Header Column Name**	**Description**
1	VERSION	Version number
2	MODEL	User provided text string designating model name
3	DESC	User provided text string describing the verification task
4	FCST_LEAD	Forecast lead time in HHMMSS format
5	FCST_VALID_BEG	Forecast valid start time in YYYYMMDD_HHMMSS format
6	FCST_VALID_END	Forecast valid end time in YYYYMMDD_HHMMSS format
7	OBS_LEAD	Observation lead time in HHMMSS format
8	OBS_VALID_BEG	Observation valid start time in YYYYMMDD_HHMMSS format
9	OBS_VALID_END	Observation valid end time in YYYYMMDD_HHMMSS format
10	FCST_VAR	Model variable
11	FCST_UNITS	Units for model variable
12	FCST_LEV	Selected Vertical level for forecast
13	OBS_VAR	Observation variable
14	OBS_UNITS	Units for observation variable
15	OBS_LEV	Selected Vertical level for observations
16	OBTYPE	Observation message type selected
17	VX_MASK	Verifying masking region indicating the masking grid or polyline region applied
18	INTERP_MTHD	Interpolation method applied to forecasts
19	INTERP_PNTS	Number of points used in interpolation method
20	FCST_THRESH	The threshold applied to the forecast
21	OBS_THRESH	The threshold applied to the observations
22	COV_THRESH	NA in Point-Stat
23	ALPHA	Error percent value used in confidence intervals
24	LINE_TYPE	Output line types are listed in tables 7.2 through 7.21.

Table 7.2: Format information for FHO (Forecast, Hit rate, Observation rate) output line type.

FHO OUTPUT FORMAT		
Column Number	**FHO Column Name**	**Description**
24	FHO	Forecast, Hit, Observation line type
25	TOTAL	Total number of matched pairs
26	F_RATE	Forecast rate
27	H_RATE	Hit rate
28	O_RATE	Observation rate

Table 7.3: Format information for CTC (Contingency Table Counts) output line type.

CTC OUTPUT FORMAT		
Column Number	**CTC Column Name**	**Description**
24	CTC	Contingency Table Counts line type
25	TOTAL	Total number of matched pairs
26	FY_OY	Number of forecast yes and observation yes
27	FY_ON	Number of forecast yes and observation no
28	FN_OY	Number of forecast no and observation yes
29	FN_ON	Number of forecast no and observation no

Table 7.4: Format information for CTS (Contingency Table Statistics) output line type.

CTS OUTPUT FORMAT		
Column Number	**CTS Column Name**	**Description**
24	CTS	Contingency Table Statistics line type
25	TOTAL	Total number of matched pairs
26-30	BASER, BASER_NCL, BASER_NCU, BASER_BCL, BASER_BCU	Base rate including normal and bootstrap upper and lower confidence limits
31-35	FMEAN, FMEAN_NCL, FMEAN_NCU, FMEAN_BCL, FMEAN_BCU	Forecast mean including normal and bootstrap upper and lower confidence limits
36-40	ACC, ACC_NCL, ACC_NCU, ACC_BCL, ACC_BCU	Accuracy including normal and bootstrap upper and lower confidence limits
41-43	FBIAS, FBIAS_BCL, FBIAS_BCU	Frequency Bias including bootstrap upper and lower confidence limits
44-48	PODY, PODY_NCL, PODY_NCU, PODY_BCL, PODY_BCU	Probability of detecting yes including normal and bootstrap upper and lower confidence limits
49-53	PODN, PODN_NCL, PODN_NCU, PODN_BCL, PODN_BCU	Probability of detecting no including normal and bootstrap upper and lower confidence limits
54-58	POFD, POFD_NCL, POFD_NCU, POFD_BCL, POFD_BCU	Probability of false detection including normal and bootstrap upper and lower confidence limits
59-63	FAR, FAR_NCL, FAR_NCU, FAR_BCL, FAR_BCU	False alarm ratio including normal and bootstrap upper and lower confidence limits
64-68	CSI, CSI_NCL, CSI_NCU, CSI_BCL, CSI_BCU	Critical Success Index including normal and bootstrap upper and lower confidence limits
69-71	GSS, GSS_BCL, GSS_BCU	Gilbert Skill Score including bootstrap upper and lower confidence limits

Table 7.5: Format information for CTS (Contingency Table Statistics) output line type, continued from above

CTS OUTPUT FORMAT		
Column Number	**CTS Column Name**	**Description**
72-76	HK, HK_NCL, HK_NCU, HK_BCL, HK_BCU	Hanssen-Kuipers Discriminant including normal and bootstrap upper and lower confidence limits
77-79	HSS, HSS_BCL, HSS_BCU	Heidke Skill Score including bootstrap upper and lower confidence limits
80-84	ODDS, ODDS_NCL, ODDS_NCU, ODDS_BCL, ODDS_BCU	Odds Ratio including normal and bootstrap upper and lower confidence limits
85-89	LODDS, LODDS_NCL, LODDS_NCU, LODDS_BCL, LODDS_BCU	Logarithm of the Odds Ratio including normal and bootstrap upper and lower confidence limits
90-94	ORSS, ORSS _NCL, ORSS _NCU, ORSS _BCL, ORSS _BCU	Odds Ratio Skill Score including normal and bootstrap upper and lower confidence limits
95-99	EDS, EDS _NCL, EDS _NCU, EDS _BCL, EDS _BCU	Extreme Dependency Score including normal and bootstrap upper and lower confidence limits
100-104	SEDS, SEDS _NCL, SEDS _NCU, SEDS _BCL, SEDS _BCU	Symmetric Extreme Dependency Score including normal and bootstrap upper and lower confidence limits
105-109	EDI, EDI _NCL, EDI _NCU, EDI _BCL, EDI _BCU	Extreme Dependency Index including normal and bootstrap upper and lower confidence limits
111-113	SEDI, SEDI _NCL, SEDI _NCU, SEDI _BCL, SEDI _BCU	Symmetric Extremal Depenency Index including normal and bootstrap upper and lower confidence limits
115-117	BAGSS, BAGSS_BCL, BAGSS_BCU	Bias Adjusted Gilbert Skill Score including bootstrap upper and lower confidence limits

Table 7.6: Format information for CNT(Continuous Statistics) output line type.

CNT OUTPUT FORMAT		
Column Number	**CNT Column Name**	**Description**
24	CNT	Continuous statistics line type
25	TOTAL	Total number of matched pairs
26-30	FBAR, FBAR_NCL, FBAR_NCU, FBAR_BCL, FBAR_BCU	Forecast mean including normal and bootstrap upper and lower confidence limits
31-35	FSTDEV, FSTDEV_NCL, FSTDEV_NCU, FSTDEV_BCL, FSTDEV_BCU	Standard deviation of the forecasts including normal and bootstrap upper and lower confidence limits
36-40	OBAR, OBAR_NCL, OBAR_NCU, OBAR_BCL, OBAR_BCU	Observation mean including normal and bootstrap upper and lower confidence limits
41-45	OSTDEV, OSTDEV_NCL, OSTDEV_NCU, OSTDEV_BCL, OSTDEV_BCU	Standard deviation of the observations including normal and bootstrap upper and lower confidence limits
46-50	PR_CORR, PR_CORR_NCL, PR_CORR_NCU, PR_CORR_BCL, PR_CORR_BCU	Pearson correlation coefficient including normal and bootstrap upper and lower confidence limits
51	SP_CORR	Spearman's rank correlation coefficient
52	KT_CORR	Kendall's tau statistic
53	RANKS	Number of ranks used in computing Kendall's tau statistic
54	FRANK_TIES	Number of tied forecast ranks used in computing Kendall's tau statistic
55	ORANK_TIES	Number of tied observation ranks used in computing Kendall's tau statistic
56-60	ME, ME_NCL, ME_NCU, ME_BCL, ME_BCU	Mean error (F-O) including normal and bootstrap upper and lower confidence limits
61-65	ESTDEV, ESTDEV_NCL, ESTDEV_NCU, ESTDEV_BCL, ESTDEV_BCU	Standard deviation of the error including normal and bootstrap upper and lower confidence limits

Table 7.7: Format information for CNT(Continuous Statistics) output line type continued from above table

CNT OUTPUT FORMAT		
Column Number	**CNT Column Name**	**Description**
66-68	MBIAS, MBIAS_BCL, MBIAS_BCU	Multiplicative bias including bootstrap upper and lower confidence limits
69-71	MAE, MAE_BCL, MAE_BCU	Mean absolute error including bootstrap upper and lower confidence limits
72-74	MSE, MSE_BCL, MSE_BCU	Mean squared error including bootstrap upper and lower confidence limits
75-77	BCMSE, BCMSE_BCL, BCMSE_BCU	Bias-corrected mean squared error including bootstrap upper and lower confidence limits
78-80	RMSE, RMSE_BCL, RMSE_BCU	Root mean squared error including bootstrap upper and lower confidence limits
81-94	E10, E10_BCL, E10_BCU, E25, E25_BCL, E25_BCU, E50, E50_BCL, E50_BCU, E75, E75_BCL, E75_BCU, E90, E90_BCL, E90_BCU	10th, 25th, 50th, 75th, and 90th percentiles of the error including bootstrap upper and lower confidence limits
96-98	IQR, IQR _BCL, IQR _BCU	The Interquartile Range including bootstrap upper and lower confidence limits
99-101	MAD, MAD_BCL, MAD_BCU	The Median Absolute Deviation including bootstrap upper and lower confidence limits
102-106	ANOM_CORR, ANOM_CORR_NCL, ANOM_CORR_NCU, ANOM_CORR_BCL, ANOM_CORR_BCU	The Anomaly Correlation including normal and bootstrap upper and lower confidence limits
107-109	ME2, ME2_BCL, ME2_BCU	The square of the mean error (bias) including bootstrap upper and lower confidence limits
110-112	MSESS, MSESS_BCL, MSESS_BCU	The mean squared error skill score including bootstrap upper and lower confidence limits
113-115	RMSFA, RMSFA_BCL, RMSFA_BCU	Root mean squared forecast anomaly (f-c) including bootstrap upper and lower confidence limits
116-118	RMSOA, RMSOA_BCL, RMSOA_BCU	Root mean squared observation anomaly (o-c) including bootstrap upper and lower confidence limits

Table 7.8: Format information for MCTC (Multi-category Contingency Table Count) output line type.

MCTC OUTPUT FORMAT		
Column Number	**MCTC Column Name**	**Description**
24	MCTC	Multi-category Contingency Table Counts line type
25	TOTAL	Total number of matched pairs
26	N_CAT	Dimension of the contingency table
27	Fi_Oj	Count of events in forecast category i and observation category j, with the observations incrementing first (repeated)

Table 7.9: Format information for MCTS (Multi- category Contingency Table Statistics) output line type.

MCTS OUTPUT FORMAT		
Column Number	**MCTS Column Name**	**Description**
24	MCTS	Multi-category Contingency Table Statistics line type
25	TOTAL	Total number of matched pairs
26	N_CAT	The total number of categories in each of dimension of the contingency table. So the total number of cells is N_CAT*N_CAT.
27-31	ACC, ACC_NCL, ACC_NCU, ACC_BCL, ACC_BCU	Accuracy, normal confidence limits and bootstrap confidence limits
32-34	HK, HK_BCL, HK_BCU	Hanssen and Kuipers Discriminant and bootstrap confidence limits
35-37	HSS, HSS_BCL, HSS_BCU	Heidke Skill Score and bootstrap confidence limits
38-40	GER, GER_BCL, GER_BCU	Gerrity Score and bootstrap confidence limits

Table 7.10: Format information for PCT (Contingency Table Counts for Probabilistic forecasts) output line type.

PCT OUTPUT FORMAT		
Column Number	**PCT Column Name**	**Description**
24	PCT	Probability contingency table count line type
25	TOTAL	Total number of matched pairs
26	N_THRESH	Number of probability thresholds
27	THRESH_i	The ith probability threshold value (repeated)
28	OY_i	Number of observation yes when forecast is between the ith and i+1th probability thresholds (repeated)
29	ON_i	Number of observation no when forecast is between the ith and i+1th probability thresholds (repeated)
*	THRESH_n	Last probability threshold value

Table 7.11: Format information for PSTD (Contingency Table Statistics for Probabilistic forecasts) output line type.

PSTD OUTPUT FORMAT		
Column Number	**PSTD Column Name**	**Description**
24	PSTD	Probabilistic statistics for dichotomous outcome line type
25	TOTAL	Total number of matched pairs
26	N_THRESH	Number of probability thresholds
27-29	BASER, BASER_NCL, BASER_NCU	The Base Rate, including normal upper and lower confidence limits
30	RELIABILITY	Reliability
31	RESOLUTION	Resolution
32	UNCERTAINTY	Uncertainty
33	ROC_AUC	Area under the receiver operating characteristic curve
34-36	BRIER, BRIER_NCL, BRIER_NCU	Brier Score including normal upper and lower confidence limits
37-39	BRIERCL, BRIERCL_NCL, BRIERCL_NCU	Climatological Brier Score including upper and lower normal confidence limits
40	BSS	Brier Skill Score relative to external climatology
41	BSS_SMPL	Brier Skill Score relative to sample climatology
42	THRESH_i	The ith probability threshold value (repeated)

Table 7.12: Format information for PJC (Joint and Conditional factorization for Probabilistic forecasts) output line type.

PJC OUTPUT FORMAT		
Column Number	**PJC Column Name**	**Description**
24	PJC	Probabilistic Joint/Continuous line type
25	TOTAL	Total number of matched pairs
26	N_THRESH	Number of probability thresholds
27	THRESH_i	The ith probability threshold value (repeated)
28	OY_TP_i	Number of observation yes when forecast is between the ith and i+1th probability thresholds as a proportion of the total OY (repeated)
29	ON_TP_i	Number of observation no when forecast is between the ith and i+1th probability thresholds as a proportion of the total ON (repeated)
30	CALIBRATION_i	Calibration when forecast is between the ith and i+1th probability thresholds (repeated)
31	REFINEMENT_i	Refinement when forecast is between the ith and i+1th probability thresholds (repeated)
32	LIKELIHOOD_i	Likelihood when forecast is between the ith and i+1th probability thresholds (repeated
33	BASER_i	Base rate when forecast is between the ith and i+1th probability thresholds (repeated)
*	THRESH_n	Last probability threshold value

Table 7.13: Format information for PRC (PRC for Receiver Operating Characteristic for Probabilistic forecasts) output line type.

PRC OUTPUT FORMAT		
Column Number	**PRC Column Name**	**Description**
24	PRC	Probability ROC points line type
25	TOTAL	Total number of matched pairs
26	N_THRESH	Number of probability thresholds
27	THRESH_i	The ith probability threshold value (repeated)
28	PODY_i	Probability of detecting yes when forecast is greater than the ith probability thresholds (repeated)
29	POFD_i	Probability of false detection when forecast is greater than the ith probability thresholds (repeated)
*	THRESH_n	Last probability threshold value

Table 7.14: Format information for ECLV (ECLV for Economic Cost/Loss Relative Value) output line type.

ECLV OUTPUT FORMAT		
Column Number	**PRC Column Name**	**Description**
24	ECLV	Economic Cost/Loss Relative Value line type
25	TOTAL	Total number of matched pairs
26	BASER	Base rate
27	VALUE_BASER	Economic value of the base rate
28	N_PNT	Number of Cost/Loss ratios
29	CL_i	ith Cost/Loss ratio evaluated
30	VALUE_i	Relative value for the ith Cost/Loss ratio

Table 7.15: Format information for SL1L2 (Scalar Partial Sums) output line type.

SL1L2 OUTPUT FORMAT		
Column Number	**SL1L2 Column Name**	**Description**
24	SL1L2	Scalar L1L2 line type
25	TOTAL	Total number of matched pairs of forecast (f) and observation (o)
26	FBAR	Mean(f)
27	OBAR	Mean(o)
28	FOBAR	Mean(f*o)
29	FFBAR	Mean(f^2)
30	OOBAR	Mean(o^2)
31	MAE	Mean Absolute Error

Table 7.16: Format information for SAL1L2 (Scalar Anomaly Partial Sums) output line type.

SAL1L2 OUTPUT FORMAT		
Column Number	**SAL1L2 Column Name**	**Description**
24	SAL1L2	Scalar Anomaly L1L2 line type
25	TOTAL	Total number of matched triplets of forecast (f), observation (o), and climatological value (c)
26	FABAR	Mean(f-c)
27	OABAR	Mean(o-c)
28	FOABAR	Mean((f-c)*(o-c))
29	FFABAR	Mean($(f-c)^2$)
30	OOABAR	Mean($(o-c)^2$)
31	MAE	Mean Absolute Error

Table 7.17: Format information for VL1L2 (Vector Partial Sums) output line type.

VL1L2 OUTPUT FORMAT		
Column Number	**VL1L2 Column Name**	**Description**
24	VL1L2	Vector L1L2 line type
25	TOTAL	Total number of matched pairs of forecast winds (uf, vf) and observation winds (uo, vo)
26	UFBAR	Mean(uf)
27	VFBAR	Mean(vf)
28	UOBAR	Mean(uo)
29	VOBAR	Mean(vo)
30	UVFOBAR	Mean(uf*uo+vf*vo)
31	UVFFBAR	Mean(uf^2+vf^2)
32	UVOOBAR	Mean(uo^2+vo^2)
33	F_SPEED_BAR	Mean forecast wind speed
34	O_SPEED_BAR	Mean observed wind speed

Table 7.18: Format information for VAL1L2 (Vector Anomaly Partial Sums) output line type.

VAL1L2 OUTPUT FORMAT		
Column Number	**VAL1L2 Column Name**	**Description**
24	VAL1L2	Vector Anomaly L1L2 line type
25	TOTAL	Total number of matched triplets of forecast winds (uf, vf), observation winds (uo, vo), and climatological winds (uc, vc)
26	UFABAR	Mean(uf-uc)
27	VFABAR	Mean(vf-vc)
28	UOABAR	Mean(uo-uc)
29	VOABAR	Mean(vo-vc)
30	UVFOABAR	Mean((uf-uc)*(uo-uc)+(vf-vc)*(vo-vc))
31	UVFFABAR	Mean($(uf-uc)^2$+$(vf-vc)^2$)
32	UVOOABAR	Mean($(uo-uc)^2$+$(vo-vc)^2$)

Table 7.19: Format information for VAL1L2 (Vector Anomaly Partial Sums) output line type. Note that each statistic (except TOTAL) is followed by two columns giving bootstrap confidence intervals. These confidence intervals are not currently calculated for this release of MET, but will be in future releases.

VCNT OUTPUT FORMAT		
Column Numbers	**VCNT Column Name**	**Description**
24	VCNT	Vector Continuous Statistics line type
25	TOTAL	Total number of data points
26–28	FBAR	Mean value of forecast wind speed
29–31	OBAR	Mean value of observed wind speed
32–34	FS_RMS	Root mean square forecast wind speed
35–37	OS_RMS	Root mean square observed wind speed
38–40	MSVE	Mean squared length of the vector difference between the forecast and observed winds
41–43	RMSVE	Square root of MSVE
45–46	FSTDEV	Standard deviation of the forecast wind speed
47–49	OSTDEV	Standard deviation of the observed wind field
50–52	FDIR	Direction of the average forecast wind vector
53–55	ODIR	Direction of the average observed wind vector
56–58	FBAR_SPEED	Length (speed) of the average forecast wind vector
59–61	OBAR_SPEED	Length (speed) of the average observed wind vector
62–64	VDIFF_SPEED	Length (speed) of the vector difference between the average forecast and average observed wind vectors
65–67	VDIFF_DIR	Direction of the vector difference between the average forecast and average wind vectors
68–70	SPEED_ERR	Difference between the length of the average forecast wind vector and the average observed wind vector (in the sense F - O)
71–73	SPEED_ABSERR	Absolute value of SPEED_ERR
74–76	DIR_ERR	Signed angle between the directions of the average forecast and observed wing vectors. Positive if the forecast wind vector is counterclockwise from the observed wind vector
77–79	DIR_ABSERR	Absolute value of DIR_ABSERR

Table 7.21: Format information for MPR (Matched Pair) output line type.

MPR OUTPUT FORMAT		
Column Number	**MPR Column Name**	**Description**
24	MPR	Matched Pair line type
25	TOTAL	Total number of matched pairs
26	INDEX	Index for the current matched pair
27	OBS_SID	Station Identifier of observation
28	OBS_LAT	Latitude of the observation in degrees north
29	OBS_LON	Longitude of the observation in degrees east
30	OBS_LVL	Pressure level of the observation in hPa or accumulation interval in hours
31	OBS_ELV	Elevation of the observation in meters above sea level
32	FCST	Forecast value interpolated to the observation location
33	OBS	Observation value
34	OBS_QC	Quality control flag for observation
35	CLIMO_MEAN	Climatological mean value
36	CLIMO_STDEV	Climatological standard deviation value
37	CLIMO_CDF	Climatological cumulative distribution function value

The STAT output files described for point_stat may be used as inputs to the Stat-Analysis tool. For more information on using the Stat-Analysis tool to create stratifications and aggregations of the STAT files produced by point_stat, please see Chapter 12.

Chapter 8

Grid-Stat Tool

8.1 Introduction

The Grid-Stat tool provides verification statistics for a matched forecast and observation grid. All of the forecast grid points in the region of interest are matched to observation grid points on the same grid. All the matched grid points are used to compute the verification statistics. The Grid-Stat tool functions in much the same way as the Point-Stat tool, except that no interpolation is required because the forecasts and observations are on the same grid. However, the interpolation parameters may be used to perform a smoothing operation on the forecast and observation fields prior to verification. In addition to traditional verification approaches, the Grid-Stat tool includes Fourier decompositions, gradient statistics, distance metrics, and neighborhood methods, designed to examine forecast performance as a function of spatial scale.

Scientific and statistical aspects of the Grid-Stat tool are briefly described in this chapter, followed by practical details regarding usage and output from the tool.

8.2 Scientific and statistical aspects

8.2.1 Statistical measures

The Grid-Stat tool computes a wide variety of verification statistics. Broadly speaking, these statistics can be subdivided into three types of statistics: measures for categorical variables, measures for continuous variables, and measures for probabilistic forecasts. Further, when a climatology file is included, reference statistics for the forecasts compared to the climatology can be calculated. These categories of measures are briefly described here; specific descriptions of all measures are provided in Appendix C. Additional information can be found in Wilks (2011) and Jolliffe and Stephenson (2012), and on the world-wide web at

`http://www.cawcr.gov.au/projects/verification/verif_web_page.html`.

In addition to these verification measures, the Grid-Stat tool also computes partial sums and other FHO statistics that are produced by the NCEP verification system. These statistics are also described in Appendix C.

Measures for categorical variables

Categorical verification statistics are used to evaluate forecasts that are in the form of a discrete set of categories rather than on a continuous scale. Grid-Stat computes both 2x2 and multi-category contingency tables and their associated statistics, similar to Point-Stat. See Appendix C for more information.

Measures for continuous variables

For continuous variables, many verification measures are based on the forecast error (i.e., f - o). However, it also is of interest to investigate characteristics of the forecasts, and the observations, as well as their relationship. These concepts are consistent with the general framework for verification outlined by Murphy and Winkler (1987). The statistics produced by MET for continuous forecasts represent this philosophy of verification, which focuses on a variety of aspects of performance rather than a single measure. See Appendix C for specific information.

A user may wish to eliminate certain values of the forecasts from the calculation of statistics, a process referred to here as "conditional verification". For example, a user may eliminate all temperatures above freezing and then calculate the error statistics only for those forecasts of below freezing temperatures. Another common example involves verification of wind forecasts. Since wind direction is indeterminate at very low wind speeds, the user may wish to set a minimum wind speed threshold prior to calculating error statistics for wind direction. The user may specify these threhsolds in the configuration file to specify the conditional verification. Thresholds can be specified using the usual Fortran conventions (<, <=, ==, !-, >=, or >) followed by a numeric value. The threshold type may also be specified using two letter abbreviations (lt, le, eq, ne, ge, gt). Further, more complex thresholds can be achieved by defining multiple thresholds and using && or || to string together event definition logic. The forecast and observation threshold can be used together according to user preference by specifying one of: UNION, INTERSECTION, or SYMDIFF (symmetric difference).

Measures for probabilistic forecasts and dichotomous outcomes

For probabilistic forecasts, many verification measures are based on reliability, accuracy and bias. However, it also is of interest to investigate joint and conditional distributions of the forecasts and the observations, as in Wilks (2011). See Appendix C for specific information.

Probabilistic forecast values are assumed to have a range of either 0 to 1 or 0 to 100. If the max data value is > 1, we assume the data range is 0 to 100, and divide all the values by 100. If the max data value is $<= 1$, then we use the values as is. Further, thresholds are applied to the probabilities with equality on the

lower end. For example, with a forecast probability p, and thresholds t1 and t2, the range is defined as: t1 <= p < t2. The exception is for the highest set of thresholds, when the range includes 1: t1 <= p <= 1.To make configuration easier, in METv6.0, these probabilities may be specified in the configuration file as a list (>0.00,>0.25,>0.50,>0.75,>1.00) or using shorthand notation (==0.25) for bins of equal width.

In METv6.0, when the "prob" entry is set as a dictionary to define the field of interest, setting "prob_as_scalar = TRUE" indicates that this data should be processed as regular scalars rather than probabilities.For example, this option can be used to compute traditional 2x2 contingency tables and neighborhood verification statistics for probability data. It can also be used to compare two probability fields directly.

Use of a climatology field for comparative verification

The Grid-Stat tool allows evaluation of model forecasts compared with a user-supplied climatology. Prior to calculation of statistics, the climatology must be put on the same grid as the forecasts and observations. In particular, the anomaly correlation and mean squared error skill score provide a measure of the forecast skill versus the climatology. For more details about climatological comparisons and reference forecasts, see the relevant section in the Point-Stat Chapter, Section 7.2.3

Use of analysis fields for verification

The Grid-Stat tool allows evaluation of model forecasts using model analysis fields. However, users are cautioned that an analysis field is not independent of its parent model; for this reason verification of model output using an analysis field from the same model is generally not recommended and is not likely to yield meaningful information about model performance.

8.2.2 Statistical confidence intervals

The confidence intervals for the Grid-Stat tool are the same as those provided for the Point-Stat tool except that the scores are based on pairing grid points with grid points so that there are likely more values for each field making any assumptions based on the central limit theorem more likely to be valid. However, it should be noted that spatial (and temporal) correlations are not presently taken into account in the confidence interval calculations. Therefore, confidence intervals reported may be somewhat too narrow (e.g., Efron 2007). See Appendix D for details regarding confidence intervals provided by MET.

8.2.3 Grid weighting

When computing continuous statistics on a regular large scale or global latitude-longitude grid, weighting may be applied in order to compensate for the meridian convergence toward higher latitudes. Grid square area weighting or weighting based on the cosine of the latitude are two configuration options in both point-stat and grid-stat. See 3.5.1for more information.

8.2.4 Neighborhood methods

MET also incorporates several neighborhood methods to give credit to forecasts that are close to the observations, but not necessarily exactly matched up in space. Also referred to as "fuzzy" verification methods, these methods do not just compare a single forecast at each grid point to a single observation at each grid point; they compare the forecasts and observations in a neighborhood surrounding the point of interest. With the neighborhood method, the user chooses a distance within which the forecast event can fall from the observed event and still be considered a hit. In MET this is implemented by defining a square search window around each grid point. Within the search window, the number of observed events is compared to the number of forecast events. In this way, credit is given to forecasts that are close to the observations without requiring a strict match between forecasted events and observed events at any particular grid point. The neighborhood methods allow the user to see how forecast skill varies with neighborhood size and can help determine the smallest neighborhood size that can be used to give sufficiently accurate forecasts.

There are several ways to present the results of the neighborhood approaches, such as the Fractions Skill Score (FSS) or the Fractions Brier Score (FBS). These scores are presented in Appendix C. One can also simply up-scale the information on the forecast verification grid by smoothing or resampling within a specified neighborhood around each grid point and recalculate the traditional verification metrics on the coarser grid. The MET output includes traditional contingency table statistics for each threshold and neighborhood window size.

The user must specify several parameters in the grid_stat configuration file to utilize the neighborhood approach, such as the interpolation method, size of the smoothing window, and required fraction of valid data points within the smoothing window. For FSS-specific results, the user must specify the size of the neighborhood window, the required fraction of valid data points within the window, and the fractional coverage threshold from which the contingency tables are defined. These parameters are described further in the practical information section below.

8.2.5 Fourier Decomposition

The MET software will compute the full one-dimensional Fourier transform, then do a partial inverse transform based on the two user-defined wave numbers. These two wave numbers define a band pass filter in the Fourier domain. This process is conceptually similar to the operation of projecting onto subspace in linear algebra. If one were to sum up all possible wave numbers the result would be to simply reproduce the raw data.

Decomposition via Fourier transform allows the user to evaluate the model separately at each spatial frequency. As an example, the Fourier analysis allows users to examine the "dieoff", or reduction, in anomaly correlation of geopotential height at various levels for bands of waves. A band of low wave numbers, say 0 - 3, represent larger frequency components, while a band of higher wave numbers, for example 70 - 72, represent smaller frequency components. Generally, anomaly correlation should be higher for frequencies with low wave numbers than for frequencies with high wave numbers, hence the "dieoff".

Wavelets, and in particular the MET wavelet tool, can also be used to define a band pass filter (Casati et al, 2004; Weniger et al 2016). Both the Fourier and wavelet methods can be used to look at different spatial scales.

8.2.6 Gradient Statistics

The S1 score has been in historical use for verification of forecasts, particularly for variables such as pressure and geopotential height. This score compares differences between adjacent grid points in the forecast and observed fields. When the adjacent points in both forecast and observed fields exhibit the same differences, the S1 score will be the perfect value of 0. Larger differences will result in a larger score.

Differences are computed in both of the horizontal grid directions and is not a true mathematical gradient. Because the S1 score focuses on differences only, any bias in the forecast will not be measured. Further, the score depends on the domain and spacing of the grid, so can only be compared on forecasts with identical grids.

8.2.7 Distance Maps

The following methods can all be computed efficiently by utilizing fast algorithms developed for calculating distance maps. A distance map results from calculating the shortest distance from every grid point, $s = (x, y)$, in the domain, D, to the nearest one-valued grid point. In each of the following, it is understood that they are calculated between event areas A, from one field and observation event areas B from another. If the measure is applied to a feature within a field, then the distance map is still calculated over the entire original domain. Some of the distance map statistics are computed over the entire distance map, while others use only parts of it.

Because these methods rely on the distance map, it is helpful to understand precisely what such maps do. Figure 8.1 demonstrates the path of the shortest distance to the nearest event point in the event area A marked by the gray rectangle in the diagram. Note that the arrows all point to a grid point on the boundary of the event area A as it would be a longer distance to any point in its interior. Figure 8.2 demonstrates the shortest distances from every grid point inside a second event area marked by the gray circle labeled B to the same event area A as in Figure 8.1. Note that all of the distances are to points on a small subsection (indicated by the yellow stretch) of the subset A.

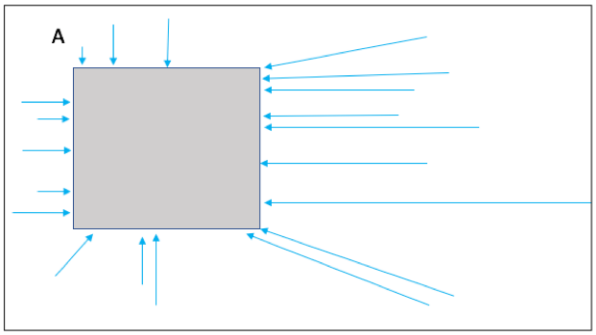

Figure 8.1: The above diagram depicts how a distance map is formed. From every grid point in the domain (depicted by the larger rectangle), the shortest distance from that grid to the nearest non-zero grid point (event; depicted by the gray rectangle labeled as A) is calculated (a sample of grid points with arrows indicate the path of the shortest distance with the length of the arrow equal to this distance. In a distance map, the value at each grid point is this distance. For example, grid points within the rectangle A will all have value zero in the distance map.

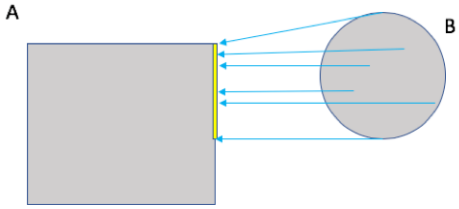

Figure 8.2: Diagram depicting the shortest distances from one event area to another. The yellow bar indicates the part of the event area A to where all of the shortest distances from B are calculated. That is, the shortest distances from every point inside the set B to the set A all point to a point along the yellow bar.

While Figure 8.1 and Figure 8.2 are helpful in illustrating the idea of a distance map, Figure 8.3 shows an actual distance map calculated for binary fields consisting of circular event areas, where one field has two circular event areas labeled A, and the second has one circular event area labeled B. Notice that the values of the distance map inside the event areas are all zero (dark blue) and the distances grow larger in the pattern of concentric circles around these event areas as grid cells move further away. Finally, Figure 8.4 depicts special situations from which the distance map measures to be discussed are calculated. In particular, the top left panel shows the absolute difference between the two distance maps presented in the bottom row of Figure 8.3. The top right panel shows the portion of the distance map for A that falls within the event area of B, and the bottom left depicts the portion of the distance map for B that falls within the event area A. That is, the first shows the shortest distances from every grid point in the set B to the nearest grid point in the event area A, and the latter shows the shortest distance from every grid point in A to the nearest grid point in B.

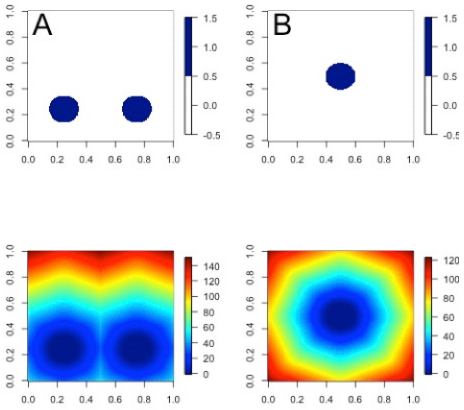

Figure 8.3: Binary fields (top) with event areas A (consisting of two circular event areas) and a second field with event area B (single circular area) with their respective distance maps (bottom).

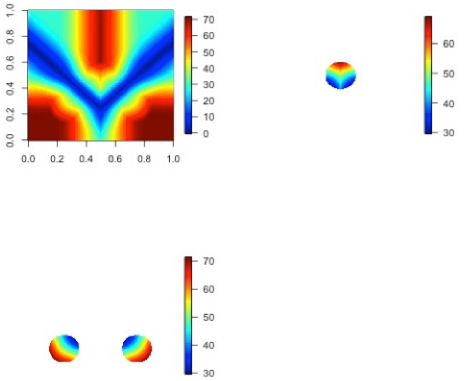

Figure 8.4: The absolute difference between the distance maps in the bottom row of Figure 8.3 (top left), the shortest distances from every grid point in B to the nearest grid point in A (top right), and the shortest distances from every grid point in A to the nearest grid points in B (bottom left). The latter two do not have axes in order to emphasize that the distances are now only considered from within the respective event sets. The top right graphic is the distance map of A conditioned on the presence of an event from B, and that in the bottom left is the distance map of B conditioned on the presence of an event from A.

The statistics derived from these distance maps are described in Appendix C.6. For each combination of input field and categorical threshold requested in the configuration file, Grid-Stat applies that threshold to define events in the forecast and observation fields and computes distance maps for those binary fields. Statistics for all requested masking regions are derived from those distance maps. Note that the distance maps are computed only once over the full verification domain, not separately for each masking region. Events occurring outside of a masking region can affect the distance map values inside that masking region and, therefore, can also affect the distance maps statistics for that region.

8.3 Practical information

This section contains information about configuring and running the Grid-Stat tool. The Grid-Stat tool verifies gridded model data using gridded observations. The input gridded model and observation datasets must be in one of the MET supported file formats. The requirement of having all gridded fields using the same grid specification was removed in METv5.1. There is a regrid option in the configuration file that allows the user to define the grid upon which the scores will be computed. The gridded observation data may be a gridded analysis based on observations such as Stage II or Stage IV data for verifying accumulated precipitation, or a model analysis field may be used.

The Grid-Stat tool provides the capability of verifying one or more model variables/levels using multiple thresholds for each model variable/level. The Grid-Stat tool performs no interpolation when the input model, observation, and climatology datasets must be on a common grid. MET will interpolate these files to a common grid if one is specified. The interpolation parameters may be used to perform a smoothing operation on the forecast field prior to verifying it to investigate how the scale of the forecast affects the verification statistics. The Grid-Stat tool computes a number of continuous statistics for the forecast minus observation differences, discrete statistics once the data have been thresholded, or statistics for probabilistic forecasts. All types of statistics can incorporate a climatological reference.

8.3.1 grid_stat usage

The usage statement for the Grid-Stat tool is listed below:

```
Usage: grid_stat
       fcst_file
       obs_file
       config_file
       [-outdir path]
       [-log file]
       [-v level]
       [-compress level]
```

grid_stat has three required arguments and accepts several optional ones.

Required arguments for grid_stat

1. The **fcst_file** argument indicates the gridded file containing the model data to be verified.

2. The **obs_file** argument indicates the gridded file containing the gridded observations to be used for the verification of the model.

3. The **config_file** argument indicates the name of the configuration file to be used. The contents of the configuration file are discussed below.

Optional arguments for grid_stat

4. The **-outdir path** indicates the directory where output files should be written.

5. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

6. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

7. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

An example of the grid_stat calling sequence is listed below:

Example 1:

```
grid_stat sample_fcst.grb \
sample_obs.grb \
GridStatConfig
```

In Example 1, the Grid-Stat tool will verify the model data in the sample_fcst.grb GRIB file using the observations in the sample_obs.grb GRIB file applying the configuration options specified in the GridStatConfig file.

A second example of the grid_stat calling sequence is listed below:

Example 2:

```
grid_stat sample_fcst.nc
sample_obs.nc
GridStatConfig
```

In the second example, the Grid-Stat tool will verify the model data in the sample_fcst.nc NetCDF output of **pcp_combine**, using the observations in the sample_obs.nc NetCDF output of **pcp_combine**, and applying the configuration options specified in the GridStatConfig file. Because the model and observation files contain only a single field of accumulated precipitation, the GridStatConfig file should be configured to specify that only accumulated precipitation be verified.

8.3.2 grid_stat configuration file

The default configuration file for the Grid-Stat tool, named GridStatConfig_default, can be found in the installed **share/met/config** directory. Other versions of the configuration file are included in **scripts/config**. We recommend that users make a copy of the default (or other) configuration file prior to modifying it. The contents are described in more detail below.

Note that environment variables may be used when editing configuration files, as described in Section 4.1.2 for the PB2NC tool.

```
model          = "WRF";
desc           = "NA";
obtype         = "ANALYS";
fcst           = { ... }
obs            = { ... }
regrid         = { ... }
climo_mean     = { ... }
climo_stdev    = { ... }
climo_cdf      = { ... }
mask           = { grid = [ "FULL" ]; poly = []; }
ci_alpha       = [ 0.05 ];
boot           = { interval = PCTILE; rep_prop = 1.0; n_rep = 1000;
                   rng = "mt19937"; seed = ""; }
interp         = { field = BOTH; vld_thresh = 1.0; shape = SQUARE;
                   type = [ { method = NEAREST; width = 1; } ]; }
censor_thresh  = [];
censor_val     = [];
eclv_points    = 0.05;
rank_corr_flag = TRUE;
tmp_dir        = "/tmp";
output_prefix  = "";
version        = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
nbrhd = {
   field      = BOTH;
   vld_thresh = 1.0;
   shape      = SQUARE;
```

```
   width      = [ 1 ];
   cov_thresh = [ >=0.5 ];
}
```

The **nbrhd** dictionary contains a list of values to be used in defining the neighborhood to be used when computing neighborhood verification statistics. The neighborhood **shape** is a **SQUARE** or **CIRCLE** centered on the current point, and the **width** value specifies the width of the square or diameter of the circle as an odd integer.

The **field** entry is set to **BOTH**, **FCST**, **OBS**, or **NONE** to indicate the fields to which the fractional coverage derivation logic should be applied. This should always to be set to **BOTH** unless you have already computed the fractional coverage field(s) with numbers between 0 and 1 outside of MET.

The **vld_thresh** entry contains a number between 0 and 1. When performing neighborhood verification over some neighborhood of points the ratio of the number of valid data points to the total number of points in the neighborhood is computed. If that ratio is greater than this threshold, that value is included in the neighborhood verification. Setting this threshold to 1, which is the default, requires that the entire neighborhood must contain valid data. This variable will typically come into play only along the boundaries of the verification region chosen.

The **cov_thresh** entry contains a comma separated list of thresholds to be applied to the neighborhood coverage field. The coverage is the proportion of forecast points in the neighborhood that exceed the forecast threshold. For example, if 10 of the 25 forecast grid points contain values larger than a threshold of 2, then the coverage is $10/25 = 0.4$. If the coverage threshold is set to 0.5, then this neighborhood is considered to be a "No" forecast.

```
fourier = {
   wave_1d_beg = [ 0, 4, 10 ];
   wave_1d_end = [ 3, 9, 20 ];
}
```

The **fourier** entry is a dictionary which specifies the application of the Fourier decomposition method. It consists of two arrays of the same length which define the beginning and ending wave numbers to be included. If the arrays have length zero, no Fourier decomposition is applied. For each array entry, the requested Fourier decomposition is applied to the forecast and observation fields. The beginning and ending wave numbers are indicated in the MET ASCII output files by the INTERP_MTHD column (e.g. WV1_0-3 for waves 0 to 3 or WV1_10 for only wave 10). This 1-dimensional Fourier decomposition is computed along the Y-dimension only (i.e. the columns of data). It is applied to the forecast and observation fields as well as the climatological mean field, if specified. It is only defined when each grid point contains valid data. If any input field contains missing data, no Fourier decomposition is computed. The available wave numbers start at 0 (the mean across each row of data) and end at (Nx+1)/2 (the finest level of detail), where Nx is the X-dimension of the verification grid.

The **wave_1d_beg** entry is an array of integers specifying the first wave number to be included. The **wave_1d_end** entry is an array of integers specifying the last wave number to be included.

```
grad = {
   dx = [ 1 ];
   dy = [ 1 ];
}
```

The **gradient** entry is a dictionary which specifies the number and size of gradients to be computed. The **dx** and **dy** entries specify the size of the gradients in grid units in the X and Y dimensions, respectively. **dx** and **dy** are arrays of integers (positive or negative) which must have the same length, and the GRAD output line type will be computed separately for each entry. When computing gradients, the value at the (x, y) grid point is replaced by the value at the (x+dx, y+dy) grid point minus the value at (x, y). This configuration option may be set separately in each **obs.field** entry.

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

The **distance_map** entry is a dictionary containing options related to the distance map statistics in the **DMAP** output line type. The **baddeley_p** entry is an integer specifying the exponent used in the Lp-norm when computing the Baddeley Δ metric. The **baddeley_max_dist** entry is a floating point number specifying the maximum allowable distance for each distance map. Any distances larger than this number will be reset to this constant. A value of **NA** indicates that no maximum distance value should be used. The **fom_alpha** entry is a floating point number specifying the scaling constant to be used when computing Pratt's Figure of Merit. The **zhu_weight** specifies a value between 0 and 1 to define the importance of the RMSE of the binary fields (i.e. amount of overlap) versus the mean-error distance (MED). The default value of 0.5 gives equal weighting. This configuration option may be set separately in each **obs.field** entry.

```
output_flag = {
   fho    = BOTH;
   ctc    = BOTH;
   cts    = BOTH;
```

```
   mctc   = BOTH;
   mcts   = BOTH;
   cnt    = BOTH;
   sl1l2  = BOTH;
   sal1l2 = NONE;
   vl1l2  = BOTH;
   val1l2 = NONE;
   vcnt   = BOTH;
   pct    = BOTH;
   pstd   = BOTH;
   pjc    = BOTH;
   prc    = BOTH;
   eclv   = BOTH;
   nbrctc = BOTH;
   nbrcts = BOTH;
   nbrcnt = BOTH;
   grad   = BOTH;
   dmap   = BOTH;
}
```

The **output_flag** array controls the type of output that the Grid-Stat tool generates. Each flag corresponds to an output line type in the STAT file. Setting the flag to NONE indicates that the line type should not be generated. Setting the flag to STAT indicates that the line type should be written to the STAT file only. Setting the flag to BOTH indicates that the line type should be written to the STAT file as well as a separate ASCII file where the data are grouped by line type. These output flags correspond to the following types of output line types:

1. **FHO** for Forecast, Hit, Observation Rates
2. **CTC** for Contingency Table Counts
3. **CTS** for Contingency Table Statistics
4. **MCTC** for Multi-Category Contingency Table Counts
5. **MCTS** for Multi-Category Contingency Table Statistics
6. **CNT** for Continuous Statistics
7. **SL1L2** for Scalar L1L2 Partial Sums
8. **SAL1L2** for Scalar Anomaly L1L2 Partial Sums when climatological data is supplied
9. **VL1L2** for Vector L1L2 Partial Sums
10. **VAL1L2** for Vector Anomaly L1L2 Partial Sums when climatological data is supplied
11. **VCNT** for Vector Contingency Table Statistics

12. **PCT** for Contingency Table Counts for Probabilistic forecasts
13. **PSTD** for Contingency Table Statistics for Probabilistic forecasts
14. **PJC** for Joint and Conditional factorization for Probabilistic forecasts
15. **PRC** for Receiver Operating Characteristic for Probabilistic forecasts
16. **ECLV** for Cost/Loss Ratio Relative Value
17. **NBRCTC** for Neighborhood Contingency Table Counts
18. **NBRCTS** for Neighborhood Contingency Table Statistics
19. **NBRCNT** for Neighborhood Continuous Statistics
20. **GRAD** for Gradient Statistics
21. **DMAP** for Distance Map Statistics

Note that the first two line types are easily derived from one another. The user is free to choose which measure is most desired. The output line types are described in more detail in Section 8.3.3.

```
nc_pairs_flag = {
   latlon       = TRUE;
   raw          = TRUE;
   diff         = TRUE;
   climo        = TRUE;
   climo_cdp    = TRUE;
   weight       = FALSE;
   nbrhd        = FALSE;
   gradient     = FALSE;
   distance_map = FALSE;
   apply_mask   = TRUE;
}
```

The **nc_pairs_flag** entry may either be set to a boolean value or a dictionary specifying which fields should be written. Setting it to TRUE indicates the output NetCDF matched pairs file should be created with all available output fields, while setting all to FALSE disables its creation. This is done regardless of if **output_flag** dictionary indicates any statistics should be computed. The **latlon**, **raw**, and **diff** entries control the creation of output variables for the latitude and longitude, the raw forecast and observed fields, and the forecast minus observation difference fields. The **climo**, **weight**, and **nbrhd** entries control the creation of output variables for the climatological mean and standard deviation fields, the grid area weights applied, and the fractional coverage fields computed for neighborhood verification methods. Setting these entries to TRUE indicates that they should be written, while setting them to FALSE disables their creation.

Setting the **climo_cdp** entry to TRUE enables the creation of an output variable for each climatological distribution percentile (CDP) threshold requested in the configuration file. Note that enabling **nbrhd** output may lead to very large output files. The **gradient** entry controls the creation of output variables for the FCST and OBS gradients in the grid-x and grid-y directions. The **distance_map** entry controls the creation of output variables for the FCST and OBS distance maps for each categorical threshold. The **apply_mask** entry controls whether to create the FCST, OBS, and DIFF output variables for all defined masking regions. Setting this to TRUE will create the FCST, OBS, and DIFF output variables for all defined masking regions. Setting this to FALSE will create the FCST, OBS, and DIFF output variables for only the FULL verification domain.

```
nc_pairs_var_name = "";
```

The **nc_pairs_var_name** entry specifies a string for each verification task. This string is parsed from each **obs.field** dictionary entry and is used to construct variable names for the NetCDF matched pairs output file. The default value of an empty string indicates that the **name** and **level** strings of the input data should be used. If the input data **level** string changes for each run of Grid-Stat, using this option to define a constant string may make downstream processing more convenient.

```
nc_pairs_var_suffix = "";
```

The **nc_pairs_var_suffix** entry is similar to the **nc_pairs_var_name** entry. It is also parsed from each **obs.field** dictionary entry. However, it defines a suffix to be appended to the output variable name. This enables the output variable names to be made unique. For example, when verifying height for multiple level types but all with the same level value, use this option to customize the output variable names. This option was previously named **nc_pairs_var_str** which is now deprecated.

8.3.3 grid_stat output

grid_stat produces output in STAT and, optionally, ASCII and NetCDF formats. The ASCII output duplicates the STAT output but has the data organized by line type. The output files are written to the default output directory or the directory specified by the -outdir command line option.

The output STAT file is named using the following naming convention:

grid_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV.stat where PREFIX indicates the user-defined output prefix, HHMMSSL indicates the forecast lead time and YYYYMMDD_HHMMSSV indicates the forecast valid time.

The output ASCII files are named similarly:

grid_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV_TYPE.txt where TYPE is one of fho, ctc, cts, mctc, mcts, cnt, sl1l2, vl1l2, vcnt, pct, pstd, pjc, prc, eclv, nbrctc, nbrcts, nbrcnt, dmap, or grad to indicate the line type it contains.

The format of the STAT and ASCII output of the Grid-Stat tool are the same as the format of the STAT and ASCII output of the Point-Stat tool with the exception of the five additional line types. Please refer to the tables in Section 7.3.3 for a description of the common output STAT and optional ASCII file line types. The formats of the five additional line types for grid_stat are explained in the following tables.

Table 8.1: Header information for each file grid-stat outputs.

HEADER		
Column Number	**Header Column Name**	**Description**
1	VERSION	Version number
2	MODEL	User provided text string designating model name
3	DESC	User provided text string describing the verification task
4	FCST_LEAD	Forecast lead time in HHMMSS format
5	FCST_VALID_BEG	Forecast valid start time in YYYYMMDD_HHMMSS format
6	FCST_VALID_END	Forecast valid end time in YYYYMMDD_HHMMSS format
7	OBS_LEAD	Observation lead time in HHMMSS format
8	OBS_VALID_BEG	Observation valid start time in YYYYMMDD_HHMMSS format
9	OBS_VALID_END	Observation valid end time in YYYYMMDD_HHMMSS format
10	FCST_VAR	Model variable
11	FCST_UNITS	Units for model variable
12	FCST_LEV	Selected Vertical level for forecast
13	OBS_VAR	Observation variable
14	OBS_UNITS	Units for observation variable
15	OBS_LEV	Selected Vertical level for observations
16	OBTYPE	User provided text string designating the observation type
17	VX_MASK	Verifying masking region indicating the masking grid or polyline region applied
18	INTERP_MTHD	Interpolation method applied to forecast field
19	INTERP_PNTS	Number of points used by interpolation method
20	FCST_THRESH	The threshold applied to the forecast
21	OBS_THRESH	The threshold applied to the observations
22	COV_THRESH	Proportion of observations in specified neighborhood which must exceed obs_thresh
23	ALPHA	Error percent value used in confidence intervals
24	LINE_TYPE	Various line type options, refer to Section 7.3.3 and the tables below.

Table 8.2: Format information for NBRCTC (Neighborhood Contingency Table Counts) output line type.

NBRCTC OUTPUT FORMAT		
Column Number	**NBRCTC Column Name**	**Description**
24	NBRCTC	Neighborhood Contingency Table Counts line type
25	TOTAL	Total number of matched pairs
26	FY_OY	Number of forecast yes and observation yes
27	FY_ON	Number of forecast yes and observation no
28	FN_OY	Number of forecast no and observation yes
29	FN_ON	Number of forecast no and observation no

Table 8.3: Format information for NBRCTS (Neighborhood Contingency Table Statistics) output line type.

NBRCTS OUTPUT FORMAT		
Column Number	**NBRCTS Column Name**	**Description**
24	NBRCTS	Neighborhood Contingency Table Statistics line type
25	TOTAL	Total number of matched pairs
26-30	BASER, BASER_NCL, BASER_NCU, BASER_BCL, BASER_BCU	Base rate including normal and bootstrap upper and lower confidence limits
31-35	FMEAN, FMEAN_NCL, FMEAN_NCU, FMEAN_BCL, FMEAN_BCU	Forecast mean including normal and bootstrap upper and lower confidence limits
36-40	ACC, ACC_NCL, ACC_NCU, ACC_BCL, ACC_BCU	Accuracy including normal and bootstrap upper and lower confidence limits
41-43	FBIAS, FBIAS_BCL, FBIAS_BCU	Frequency Bias including bootstrap upper and lower confidence limits
44-48	PODY, PODY_NCL, PODY_NCU, PODY_BCL, PODY_BCU	Probability of detecting yes including normal and bootstrap upper and lower confidence limits
49-53	PODN, PODN_NCL, PODN_NCU, PODN_BCL, PODN_BCU	Probability of detecting no including normal and bootstrap upper and lower confidence limits
54-58	POFD, POFD_NCL, POFD_NCU, POFD_BCL, POFD_BCU	Probability of false detection including normal and bootstrap upper and lower confidence limits
59-63	FAR, FAR_NCL, FAR_NCU, FAR_BCL, FAR_BCU	False alarm ratio including normal and bootstrap upper and lower confidence limits
64-68	CSI, CSI_NCL, CSI_NCU, CSI_BCL, CSI_BCU	Critical Success Index including normal and bootstrap upper and lower confidence limits
69-71	GSS, GSS_BCL, GSS_BCU	Gilbert Skill Score including bootstrap upper and lower confidence limits

Table 8.4: Format information for NBRCTS (Neighborhood Contingency Table Statistics) output line type, continued from above.

Column Number	NBRCTS Column Name	Description
72-76	HK, HK_NCL, HK_NCU, HK_BCL, HK_BCU	Hanssen-Kuipers Discriminant including normal and bootstrap upper and lower confidence limits
77-79	HSS, HSS_BCL, HSS_BCU	Heidke Skill Score including bootstrap upper and lower confidence limits
80-84	ODDS, ODDS_NCL, ODDS_NCU, ODDS_BCL, ODDS_BCU	Odds Ratio including normal and bootstrap upper and lower confidence limits
85-89	LODDS, LODDS_NCL, LODDS_NCU, LODDS_BCL, LODDS_BCU	Logarithm of the Odds Ratio including normal and bootstrap upper and lower confidence limits
90-94	ORSS, ORSS _NCL, ORSS _NCU, ORSS _BCL, ORSS _BCU	Odds Ratio Skill Score including normal and bootstrap upper and lower confidence limits
95-99	EDS, EDS _NCL, EDS _NCU, EDS _BCL, EDS _BCU	Extreme Depenency Score including normal and bootstrap upper and lower confidence limits
100-104	SEDS, SEDS _NCL, SEDS _NCU, SEDS _BCL, SEDS _BCU	Symmetric Extreme Depenency Score including normal and bootstrap upper and lower confidence limits
105-109	EDI, EDI _NCL, EDI _NCU, EDI _BCL, EDI _BCU	Extreme Depenency Index including normal and bootstrap upper and lower confidence limits
110-114	SEDI, SEDI _NCL, SEDI _NCU, SEDI _BCL, SEDI _BCU	Symmetric Extremal Depenency Index including normal and bootstrap upper and lower confidence limits
115-117	BAGSS, BAGSS_BCL, BAGSS_BCU	Bias Adjusted Gilbert Skill Score including bootstrap upper and lower confidence limits

Table 8.5: Format information for NBRCNT(Neighborhood Continuous Statistics) output line type.

NBRCNT OUTPUT FORMAT		
Column Number	**NBRCNT Column Name**	**Description**
24	NBRCNT	Neighborhood Continuous statistics line type
25	TOTAL	Total number of matched pairs
26-28	FBS, FBS_BCL, FBS_BCU	Fractions Brier Score including bootstrap upper and lower confidence limits
29-31	FSS, FSS_BCL, FSS_BCU	Fractions Skill Score including bootstrap upper and lower confidence limits
32-34	AFSS, AFSS_BCL, AFSS_BCU	Asymptotic Fractions Skill Score including bootstrap upper and lower confidence limits
35-37	UFSS, UFSS_BCL, UFSS_BCU	Uniform Fractions Skill Score including bootstrap upper and lower confidence limits
38-40	F_RATE, F_RATE _BCL, F_RATE _BCU	Forecast event frequency including bootstrap upper and lower confidence limits
41-43	O_RATE, O _RATE _BCL, O _RATE _BCU	Observed event frequency including bootstrap upper and lower confidence limits

Table 8.6: Format information for GRAD (Gradient Statistics) output line type.

GRAD OUTPUT FORMAT		
Column Number	**GRAD Column Name**	**Description**
24	GRAD	Gradient Statistics line type
25	TOTAL	Total number of matched pairs
26	FGBAR	Mean of absolute value of forecast gradients
27	OGBAR	Mean of absolute value of observed gradients
28	MGBAR	Mean of maximum of absolute values of forecast and observed gradients
29	EGBAR	Mean of absolute value of forecast minus observed gradients
30	S1	S1 score
31	S1_OG	S1 score with respect to observed gradient
32	FGOG_RATIO	Ratio of forecast and observed gradients
33	DX	Gradient size in the X-direction
34	DY	Gradient size in the Y-direction

Table 8.7: Format information for DMAP (Distance Map) output line type.

DMAP OUTPUT FORMAT		
Column Number	**DMAP Column Name**	**Description**
24	DMAP	Distance Map line type
25	TOTAL	Total number of matched pairs
26	FY	Number of forecast events
27	OY	Number of observation events
28	FBIAS	Frequency Bias
29	BADDELEY	Baddeley's Δ Metric
30	HAUSDORFF	Hausdorff Distance
31	MED_FO	Mean-error Distance from observation to forecast
32	MED_OF	Mean-error Distance from forecast to observation
33	MED_MIN	Minimum of MED_FO and MED_OF
34	MED_MAX	Maximum of MED_FO and MED_OF
35	MED_MEAN	Mean of MED_FO and MED_OF
36	FOM_FO	Pratt's Figure of Merit from observation to forecast
37	FOM_OF	Pratt's Figure of Merit from forecast to observation
38	FOM_MIN	Minimum of FOM_FO and FOM_OF
39	FOM_MAX	Maximum of FOM_FO and FOM_OF
40	FOM_MEAN	Mean of FOM_FO and FOM_OF
41	ZHU_FO	Zhu's Measure from observation to forecast
42	ZHU_OF	Zhu's Measure from forecast to observation
43	ZHU_MIN	Minimum of ZHU_FO and ZHU_OF
44	ZHU_MAX	Maximum of ZHU_FO and ZHU_OF
45	ZHU_MEAN	Mean of ZHU_FO and ZHU_OF

If requested using the **nc_pairs_flag** dictionary in the configuration file, a NetCDF file containing the matched pair and forecast minus observation difference fields for each combination of variable type/level and masking region applied will be generated. The contents of this file are determined by the contents of the nc_pairs_flag dictionary. The output NetCDF file is named similarly to the other output files: **grid_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV_pairs.nc**. Commonly available NetCDF utilities such as ncdump or ncview may be used to view the contents of the output file.

The output NetCDF file contains the dimensions and variables shown in the following Tables 8.8 and 8.9.

Table 8.8: Dimensions defined in NetCDF matched pair output.

grid_stat NETCDF DIMENSIONS	
NetCDF Dimension	**Description**
Lat	Dimension of the latitude (i.e. Number of grid points in the North-South direction)
Lon	Dimension of the longitude (i.e. Number of grid points in the East-West direction)

Table 8.9: A selection of variables that can appear in the NetCDF matched pair output.

grid_stat NETCDF VARIABLES		
NetCDF Variable	**Dimension**	**Description**
FCST_VAR_LVL_MASK _INTERP_MTHD _INTERP_PNTS	lat, lon	For each model variable (VAR), vertical level (LVL), masking region (MASK), and, if applicable, smoothing operation (INTERP_MTHD and INTERP_PNTS), the forecast value is listed for each point in the mask.
OBS_VAR_LVL_MASK	lat, lon	For each model variable (VAR), vertical level (LVL), and masking region (MASK), the observation value is listed for each point in the mask .
DIFF_FCSTVAR _FCSTLVL _OBSVAR _OBSLVL_MASK _INTERP_MTHD _INTERP_PNTS	lat, lon	For each model variable (VAR), vertical level (LVL), masking region (MASK), and, if applicable, smoothing operation (INTERP_MTHD and INTERP_PNTS), the difference (forecast - observation) is computed for each point in the mask.
FCST_XGRAD_DX FCST_YGRAD_DX OBS_XGRAD_DY OBS_YGRAD_DY	lat, lon	List the gradient of the forecast and observation fields computed in the grid-x and grid-y directions where DX and DY indicate the gradient direction and size.

The STAT output files described for **grid_stat** may be used as inputs to the Stat-Analysis tool. For more information on using the Stat-Analysis tool to create stratifications and aggregations of the STAT files produced by **grid_stat**, please see Chapter 12.

Chapter 9

Ensemble-Stat Tool

9.1 Introduction

The Ensemble-Stat tool may be run to create simple ensemble forecasts (mean, probability, spread, etc) from a set of several forecast model files to be used by the MET statistics tools. If observations are also included, ensemble statistics such as rank histograms, probability integral transform histograms, spread/skill variance, relative position and continuous ranked probability score are produced. A climatology file may also be provided, and will be used as a reference forecast in several of the output statistics. Finally, observation error perturbations can be included prior to calculation of statistics. Details about and equations for the statistics produced for ensembles are given in Appendix C C.4.

9.2 Scientific and statistical aspects

9.2.1 Ensemble forecasts derived from a set of deterministic ensemble members

Ensemble forecasts are often created as a set of deterministic forecasts. The ensemble members are rarely used separately. Instead, they can be combined in various ways to produce a forecast. MET can combine the ensemble members into some type of summary forecast according to user specifications. Ensemble means are the most common, and can be paired with the ensemble variance or spread. Maximum, minimum and other summary values are also available, with details in the practical information section.

The ensemble relative frequency is the simplest method for turning a set of deterministic forecasts into something resembling a probability forecast. MET will create the ensemble relative frequency as the proportion of ensemble members forecasting some event. For example, if 5 out of 10 ensemble members predict measurable precipitation at a grid location, then the ensemble relative frequency of precipitation will be 5/10=0.5. If the ensemble relative frequency is calibrated (unlikely) then this could be thought of as a probability of precipitation.

The neighborhood ensemble probability (NEP) and neighborhood maximum ensemble probability (NMEP) methods are described in Schwartz and Sobash (2017). They are an extension of the ensemble relative frequencies described above. The NEP value is computed by averaging the relative frequency of the event within the neighborhood over all ensemble members. The NMEP value is computed as the fraction of ensemble members for which the event is occurring somewhere within the surrounding neighborhood. The NMEP output is typically smoothed using a Gaussian kernel filter. The neighborhood sizes and smoothing options can be customized in the configuration file.

The Ensemble-Stat tool writes the gridded relative frequencies, NEP, and NMEP fields to a NetCDF output file. Probabilistic verification methods can then be applied to those fields by evaluating them with the Grid-Stat and/or Point-Stat tools.

9.2.2 Ensemble statistics

Rank histograms and probability integral transform (PIT) histograms are used to determine if the distribution of ensemble values is the same as the distribution of observed values for any forecast field (Hamill, 2001). The rank histogram is a tally of the rank of the observed value when placed in order with each of the ensemble values from the same location. If the distributions are identical, then the rank of the observation will be uniformly distributed. In other words, it will fall among the ensemble members randomly in equal likelihood. The PIT histogram applies this same concept, but transforms the actual rank into a probability to facilitate ensembles of differing sizes or with missing members.

Often, the goal of ensemble forecasting is to reproduce the distribution of observations using a set of many forecasts. In other words, the ensemble members represent the set of all possible outcomes. When this is true, the spread of the ensemble is identical to the error in the mean forecast. Though this rarely occurs in practice, the spread / skill relationship is still typically assessed for ensemble forecasts (Barker, 1991; Buizza, 1997). MET calculates the spread and skill in user defined categories according to Eckel *et al*, 2012.

The relative position (RELP) is a count of the number of times each ensemble member is closest to the observation. For stochastic or randomly derived ensembles, this statistic is meaningless. For specified ensemble members, however, it can assist users in determining if any ensemble member is performing consistently better or worse than the others.

The ranked probability score (RPS) is included in the Ranked Probability Score (RPS) line type. It is the mean of the Brier scores computed from ensemble probabilities derived for each probability category threshold (prob_cat_thresh) specified in the configuration file. The continuous ranked probability score (CRPS) is the average the distance between the forecast (ensemble) cumulative distribution function and the observation cumulative distribution function. It is an analog of the Brier score, but for continuous forecast and observation fields. (Gneiting *et al*, 2004). The CRPS statistic is included in the Ensemble Continuous Statistics (ECNT) line type, along with other statistics quantifying the ensemble spread and ensemble mean skill.

9.2.3 Ensemble observation error

In an attempt to ameliorate the effect of observation errors on the verification of forecasts, a random perturbation approach has been implemented. A great deal of user flexibility has been built in, but the methods detailed in Candile and Talagrand (2008) can be replicated using the appropriate options. The user selects a distribution for the observation error, along with parameters for that distribution. Rescaling and bias correction can also be specified prior to the perturbation. Random draws from the distribution can then be added to either, or both, of the forecast and observed fields, including ensemble members. Details about the effects of the choices on verification statistics should be considered, with many details provided in the literature (e.g. Candille and Talagrand, 2008; Saetra et al., 2004; Santos and Ghelli, 2012). Generally, perturbation make verification statistics better when applied to ensemble members, and worse when applied to the observations themselves.

Normal and uniform are common choices for the observation error distribution. The uniform distribution provides the benefit of being bounded on both sides, thus preventing the perturbation from taking on extreme values. Normal is the most common choice for observation error. However, the user should realize that with the very large samples typical in NWP, some large outliers will almost certainly be introduced with the perturbation. For variables that are bounded below by 0, and that may have inconsistent observation errors (e.g. larger errors with larger measurements), a lognormal distribution may be selected. Wind speeds and precipitation measurements are the most common of this type of NWP variable. The lognormal error perturbation prevents measurements of 0 from being perturbed, and applies larger perturbations when measurements are larger. This is often the desired behavior in these cases, but this distribution can also lead to some outliers being introduced in the perturbation step.

Observation errors differ according to instrument, temporal and spatial representation, and variable type. Unfortunately, many observation errors have not been examined or documented in the literature. Those that have usually lack information regarding their distributions and approximate parameters. Instead, a range or typical value of observation error is often reported and these are often used as an estimate of the standard deviation of some distribution. Where possible, it is recommended to use the appropriate type and size of perturbation for the observation to prevent spurious results.

9.3 Practical Information

This section contains information about configuring and running the Ensemble-Stat tool. The Ensemble-Stat tool creates or verifies gridded model data. For verification, this tool can accept either gridded or point observations. If provided, the climatology file must be gridded. The input gridded model, observation, and climatology datasets must be on the same grid prior to calculation of any statistics, and in one of the MET supported gridded file formats. If gridded files are not on the same grid, MET will do the regridding for you if you specify the desired output grid. The point observations must be formatted as the NetCDF output of the point reformatting tools described in Chapter 4.

9.3.1 ensemble_stat usage

The usage statement for the Ensemble Stat tool is shown below:

```
Usage: ensemble_stat
       n_ens ens_file_1 ... ens_file_n | ens_file_list
       config_file
       [-grid_obs file]
       [-point_obs file]
       [-ens_mean file]
       [-obs_valid_beg time]
       [-obs_valid_end time]
       [-outdir path]
       [-log file]
       [-v level]
       [-compress level]
```

ensemble_stat has three required arguments and accepts several optional ones.

Required arguments ensemble_stat

1. The **n_ens ens_file_1 ... ens_file_n** is the number of ensemble members followed by a list of ensemble member file names. This argument is not required when ensemble files are specified in the **ens_file_list**, detailed below.

2. The **ens_file_list** is an ASCII file containing a list of ensemble member file names. This is not required when a file list is included on the command line, as described above.

3. The **config_file** is an EnsembleStatConfig file containing the desired configuration settings.

Optional arguments for ensemble_stat

4. To produce ensemble statistics using gridded observations, use the **-grid_obs file** option to specify a gridded observation file. This option may be used multiple times if your observations are in several files.

5. To produce ensemble statistics using point observations, use the **-point_obs file** to specify a NetCDF point observation file. This option may be used multiple times if your observations are in several files.

6. To override the simple ensemble mean value of the input ensemble members for the ECNT, SSVAR, and ORANK line types, the **-ens_mean file** specifies an ensemble mean model data file. This option replaces the **-ssvar_mean file** from earlier versions of MET.

7. To filter point observations by time, use **-obs_valid_beg time** in YYYYMMDD[_HH[MMSS]] format to set the beginning of the matching observation time window.

8. As above, use **-obs_valid_end time** in YYYYMMDD[_HH[MMSS]] format to set the end of the matching observation time window.

9. Specify the **-outdir path** option to override the default output directory (./).

10. The **-log** file outputs log messages to the specified file.

11. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity will increase the amount of logging.

12. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

An example of the ensemble_stat calling sequence is shown below:

```
ensemble_stat \
6 sample_fcst/2009123112/*gep*/d01_2009123112_02400.grib \
config/EnsembleStatConfig \
-grid_obs sample_obs/ST4/ST4.2010010112.24h \
-point_obs out/ascii2nc/precip24_2010010112.nc \
-outdir out/ensemble_stat -v 2
```

In this example, the Ensemble-Stat tool will process six forecast files specified in the file list into an ensemble forecast. Observations in both point and grid format will be included, and used to ensemble statistics separately. Ensemble Stat will create a NetCDF file containing requested ensemble fields and an output STAT file.

9.3.2 ensemble_stat configuration file

The default configuration file for the Ensemble-Stat tool named **EnsembleStatConfig_default** can be found in the installed **share/met/config** directory. Another version is located in scripts/config. We encourage users to make a copy of these files prior to modifying their contents. Each configuration file (both the default and sample) contains many comments describing its contents. The contents of the configuration file are also described in the subsections below.

Note that environment variables may be used when editing configuration files, as described in the Section 4.1.2 for the PB2NC tool.

```
model          = "WRF";
desc           = "NA";
obtype         = "ANALYS";
regrid         = { ... }
climo_mean     = { ... }
climo_stdev    = { ... }
climo_cdf      = { ... }
obs_window     = { beg = -5400; end =  5400; }
mask           = { grid = [ "FULL" ]; poly = []; sid = []; }
ci_alpha       = [ 0.05 ];
interp         = { field = BOTH; vld_thresh = 1.0; shape = SQUARE;
                   type = [ { method = NEAREST; width = 1; } ]; }
sid_inc        = [];
sid_exc        = [];
duplicate_flag = NONE;
obs_quality    = [];
obs_summary    = NONE;
obs_perc_value = 50;
message_type_group_map = [...];
output_prefix  = "";
version        = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
ens = {
   ens_thresh = 1.0;
   vld_thresh = 1.0;
   field = [
      {
         name = "APCP";
         level = "A03";
         cat_thresh = [ >0.0, >=5.0 ];
      }
   ];
}
```

The **ens** dictionary defines which ensemble fields should be processed.

When summarizing the ensemble, compute a ratio of the number of valid ensemble fields to the total number of ensemble members. If this ratio is less than the **ens_thresh**, then quit with an error. This threshold

must be between 0 and 1. Setting this threshold to 1 will require that all ensemble members be present to be processed.

When summarizing the ensemble, for each grid point compute a ratio of the number of valid data values to the number of ensemble members. If that ratio is less than **vld_thresh**, write out bad data. This threshold must be between 0 and 1. Setting this threshold to 1 will require each grid point to contain valid data for all ensemble members.

For each **field** listed in the forecast field, give the name and vertical or accumulation level, plus one or more categorical thresholds. The thresholds are specified using symbols, as shown above. It is the user's responsibility to know the units for each model variable and to choose appropriate threshold values. The thresholds are used to define ensemble relative frequencies, e.g. a threshold of >=5 can be used to compute the proportion of ensemble members predicting precipitation of at least 5mm at each grid point.

```
nbrhd_prob = {
   width      = [ 5 ];
   shape      = CIRCLE;
   vld_thresh = 0.0;
}
```

The **nbrhd_prob** dictionary defines the neighborhoods used to compute NEP and NMEP output.

The neighborhood **shape** is a **SQUARE** or **CIRCLE** centered on the current point, and the **width** array specifies the width of the square or diameter of the circle as an odd integer. The **vld_thresh** entry is a number between 0 and 1 specifying the required ratio of valid data in the neighborhood for an output value to be computed.

If **ensemble_flag.nep** is set to TRUE, NEP output is created for each combination of the categorical threshold (**cat_thresh**) and neighborhood width specified.

```
nmep_smooth = {
   vld_thresh      = 0.0;
   shape           = CIRCLE;
   gaussian_dx     = 81.27;
   gaussian_radius = 120;
   type = [
      {
         method = GAUSSIAN;
         width  = 1;
      }
   ];
}
```

Similar to the **interp** dictionary, the **nmep_smooth** dictionary includes a **type** array of dictionaries to define one or more methods for smoothing the NMEP data. Setting the interpolation method to nearest neighbor (**NEAREST**) effectively disables this smoothing step.

If **ensemble_flag.nmep** is set to TRUE, NMEP output is created for each combination of the categorical threshold (**cat_thresh**), neighborhood width (**nbrhd_prob.width**), and smoothing method (**nmep_smooth.type**) specified.

```
obs_thresh = [ NA ];
```

The **obs_thresh** entry is an array of thresholds for filtering observation values prior to applying ensemble verification logic. The default setting of **NA** means that no observations should be filtered out. Verification output will be computed separately for each threshold specified. This option may be set separately for each **obs.field** entry.

```
skip_const = FALSE;
```

Setting **skip_const** to true tells Ensemble-Stat to exclude pairs where all the ensemble members and the observation have a constant value. For example, exclude points with zero precipitation amounts from all output line types. This option may be set separately for each **obs.field** entry. When set to false, constant points are and the observation rank is chosen at random.

```
ens_ssvar_bin_size = 1.0;
ens_phist_bin_size = 0.05;
prob_cat_thresh    = [];
```

Setting up the **fcst** and **obs** dictionaries of the configuration file is described in Section 3.5.1. The following are some special consideration for the Ensemble-Stat tool.

The **ens** and **fcst** dictionaries do not need to include the same fields. Users may specify any number of ensemble fields to be summarized, but generally there are many fewer fields with verifying observations available. The **ens** dictionary specifies the fields to be summarized while the **fcst** dictionary specifies the fields to be verified.

The **obs** dictionary looks very similar to the **fcst** dictionary. If verifying against point observations which are assigned GRIB1 codes, the observation section must be defined following GRIB1 conventions. When

verifying GRIB1 forecast data, one can easily copy over the forecast settings to the observation dictionary using **obs = fcst;**. However, when verifying non-GRIB1 forecast data, users will need to specify the **fcst** and **obs** sections separately.

The **ens_ssvar_bin_size** and **ens_phist_bin_size** specify the width of the categorical bins used to accumulate frequencies for spread-skill-variance or probability integral transform statistics, respectively.

The **prob_cat_thresh** entry is an array of thresholds to be applied in the computation of the RPS line type. Since these thresholds can change for each variable, they can be specified separately for each **fcst.field** entry. If left empty but climatology data is provided, the **climo_cdf** thresholds will be used instead. If not climatology data is provide, and the RPS output line type is requested, then the **prob_cat_thresh** array must be defined.

```
obs_error = {
   flag             = FALSE;
   dist_type        = NONE;
   dist_parm        = [];
   inst_bias_scale  = 1.0;
   inst_bias_offset = 0.0;
}
```

The **obs_error** dictionary controls how observation error information should be handled. This dictionary may be set separately for each **obs.field** entry. Observation error information can either be specified directly in the configuration file or by parsing information from an external table file. By default, the **MET_BASE/data/table_files/obs_error_table.txt** file is read but this may be overridden by setting the **$MET_OBS_ERROR_TABLE** environment variable at runtime.

The **flag** entry toggles the observation error logic on (**TRUE**) and off (**FALSE**). When flag is TRUE, random observation error perturbations are applied to the ensemble member values. No perturbation is applied to the observation values but the bias scale and offset values, if specified, are applied.

The **dist_type** entry may be set to **NONE**, **NORMAL**, **LOGNORMAL**, **EXPONENTIAL**, **CHISQUARED**, **GAMMA**, **UNIFORM**, or **BETA**. The default value of **NONE** indicates that the observation error table file should be used rather than the configuration file settings.

The **dist_parm** entry is an array of length 1 or 2 specifying the parameters for the distribution selected in **dist_type**. The **GAMMA**, **UNIFORM**, and **BETA** distributions are defined by two parameters, specified as a comma-separated list (a,b), whereas all other distributions are defined by a single parameter.

The **inst_bias_scale** and **inst_bias_offset** entries specify bias scale and offset values that should be applied to observation values prior to perturbing them. These entries enable bias-correction on the fly.

Defining the observation error information in the configuration file is convenient but limited. The random perturbations for all points in the current verification task are drawn from the same distribution. Specifying an observation error table file instead (by setting **dist_type = NONE;**) provides much finer control, enabling the user to define observation error distribution information and bias-correction logic separately for each observation variable name, message type, PrepBUFR report type, input report type, instrument type, station ID, range of heights, range of pressure levels, and range of values.

```
output_flag = {
   ecnt  = NONE;
   rps   = NONE;
   rhist = NONE;
   phist = NONE;
   orank = NONE;
   ssvar = NONE;
   relp  = NONE;
}
```

The **output_flag** array controls the type of output that is generated. Each flag corresponds to an output line type in the STAT file. Setting the flag to NONE indicates that the line type should not be generated. Setting the flag to STAT indicates that the line type should be written to the STAT file only. Setting the flag to BOTH indicates that the line type should be written to the STAT file as well as a separate ASCII file where the data is grouped by line type. The output flags correspond to the following output line types:

1. **ECNT** for Continuous Ensemble Statistics
2. **RPS** for Ranked Probability Score Statistics
3. **RHIST** for Ranked Histogram Counts
4. **PHIST** for Probability Integral Transform Histogram Counts
5. **ORANK** for Ensemble Matched Pair Information when point observations are supplied
6. **SSVAR** for Binned Spread/Skill Variance Information
7. **RELP** for Relative Position Counts

```
ensemble_flag = {
   latlon    = TRUE;
   mean      = TRUE;
   stdev     = TRUE;
```

```
   minus     = TRUE;
   plus      = TRUE;
   min       = TRUE;
   max       = TRUE;
   range     = TRUE;
   vld_count = TRUE;
   frequency = TRUE;
   nep       = FALSE;
   nmep      = FALSE;
   rank      = TRUE;
   weight    = FALSE;
}
```

The **ensemble_flag** specifies which derived ensemble fields should be calculated and output. Setting the flag to TRUE produces output of the specified field, while FALSE produces no output for that field type. The flags correspond to the following output line types:

1. Grid Latitude and Longitude Fields
2. Ensemble Mean Field
3. Ensemble Standard Deviation Field
4. Ensemble Mean - One Standard Deviation Field
5. Ensemble Mean + One Standard Deviation Field
6. Ensemble Minimum Field
7. Ensemble Maximum Field
8. Ensemble Range Field
9. Ensemble Valid Data Count
10. Ensemble Relative Frequency for each categorical threshold (**cat_thresh**) specified. This is an uncalibrated probability forecast.
11. Neighborhood Ensemble Probability for each categorical threshold (**cat_thresh**) and neighborhood width (**nbrhd_prob.width**) specified.
12. Neighborhood Maximum Ensemble Probability for each categorical threshold (**cat_thresh**), neighborhood width (**nbrhd_prob.width**), and smoothing method (**nmep_smooth.type**) specified.
13. Observation Ranks for input gridded observations are written to a separate NetCDF output file.
14. The grid area weights applied are written to the Observation Rank output file.

```
nc_var_str = "";
```

The **nc_var_str** entry specifies a string for each ensemble field and verification task. This string is parsed from each **ens.field** and **obs.field** dictionary entry and is used to customize the variable names written to theNetCDF output file. The default is an empty string, meaning that no customization is applied to the output variable names. When the Ensemble-Stat config file contains two fields with the same name and level value, this entry is used to make the resulting variable names unique.

```
rng = {
   type = "mt19937";
   seed = "";
}
```

The **rng** group defines the random number generator **type** and **seed** to be used. In the case of a tie when determining the rank of an observation, the rank is randomly chosen from all available possibilities. The randomness is determined by the random number generator specified.

The **seed** variable may be set to a specific value to make the assignment of ranks fully repeatable. When left empty, as shown above, the random number generator seed is chosen automatically which will lead to slightly different bootstrap confidence intervals being computed each time the data is run.

Refer to the description of the **boot** entry in Section 3.5.1 for more details on the random number generator.

9.3.3 ensemble_stat output

ensemble_stat can produce output in STAT, ASCII, and NetCDF formats. The ASCII output duplicates the STAT output but has the data organized by line type. The output files are written to the default output directory or the directory specified by the -outdir command line option.

The output STAT file is named using the following naming convention:

ensemble_stat_PREFIX_YYYYMMDD_HHMMSSV.stat where PREFIX indicates the user-defined output prefix and YYYYMMDD_HHMMSSV indicates the forecast valid time. Note that the forecast lead time is not included in the output file names since it would not be well-defined for time-lagged ensembles. When verifying multiple lead times for the same valid time, users should either write the output to separate directories or specify a output prefix to ensure unique file names.

The output ASCII files are named similarly:

ensemble_stat_PREFIX_YYYYMMDD_HHMMSSV_TYPE.txt where TYPE is one of ecnt, rps, rhist, phist, relp, orank, and ssvar to indicate the line type it contains.

When fields are requested in the ens dictionary of the configuration file or verification against gridded fields is performed, ensemble_stat can produce output NetCDF files using the following naming convention:

ensemble_stat_PREFIX_YYYYMMDD_HHMMSSV_TYPE.nc where TYPE is either ens or orank. The orank NetCDF output file contains gridded fields of observation ranks when the -grid_obs command line option is used. The ens NetCDF output file contains ensemble products derived from the fields requested in the ens dictionary of the configuration file. The Ensemble-Stat tool can calculate any of the following fields from the input ensemble members, as specified in the ensemble_flag dictionary in the configuration file:

Ensemble Mean fields

Ensemble Standard Deviation fields

Ensemble Mean - 1 Standard Deviation fields

Ensemble Mean + 1 Standard Deviation fields

Ensemble Minimum fields

Ensemble Maximum fields

Ensemble Range fields

Ensemble Valid Data Count fields

Ensemble Relative Frequency by threshold fields (e.g. ensemble probabilities)

Neighborhood Ensemble Probability and Neighborhood Maximum Ensemble Probability

Rank for each Observation Value (if gridded observation field provided)

When gridded or point observations are provided, using the -grid_obs and -point_obs command line options, respectively, the Ensemble-Stat tool can compute the following statistics for the fields specified in the fcst and obs dictionaries of the configuration file:

Continuous Ensemble Statistics

Ranked Histograms

Probability Integral Transform (PIT) Histograms

Relative Position Histograms

Spread/Skill Variance

Ensemble Matched Pair information

The format of the STAT and ASCII output of the Ensemble-Stat tool are described below.

Table 9.1: Header information for each file ensemble-stat outputs

HEADER		
Column Number	**Header Column Name**	**Description**
1	VERSION	Version number
2	MODEL	User provided text string designating model name
3	DESC	User provided text string describing the verification task
4	FCST_LEAD	Forecast lead time in HHMMSS format
5	FCST_VALID_BEG	Forecast valid start time in YYYYMMDD_HHMMSS format
6	FCST_VALID_END	Forecast valid end time in YYYYMMDD_HHMMSS format
7	OBS_LEAD	Observation lead time in HHMMSS format
8	OBS_VALID_BEG	Observation valid start time in YYYYMMDD_HHMMSS format
9	OBS_VALID_END	Observation valid end time in YYYYMMDD_HHMMSS format
10	FCST_VAR	Model variable
11	FCST_UNITS	Units for model variable
12	FCST_LEV	Selected Vertical level for forecast
13	OBS_VAR	Observation variable
14	OBS_UNITS	Units for observation variable
15	OBS_LEV	Selected Vertical level for observations
16	OBTYPE	Type of observation selected
17	VX_MASK	Verifying masking region indicating the masking grid or polyline region applied
18	INTERP_MTHD	Interpolation method applied to forecasts
19	INTERP_PNTS	Number of points used in interpolation method
20	FCST_THRESH	The threshold applied to the forecast
21	OBS_THRESH	The threshold applied to the observations
22	COV_THRESH	The minimum fraction of valid ensemble members required to calculate statistics.
23	ALPHA	Error percent value used in confidence intervals
24	LINE_TYPE	Output line types are listed in tables 9.4 through 9.8.

Table 9.2: Format information for ECNT (Ensemble Continuous Statistics) output line type.

ECNT OUTPUT FORMAT		
Column Number	**ECNT Column Name**	**Description**
24	ECNT	Ensemble Continuous Statistics line type
25	TOTAL	Count of observations
26	N_ENS	Number of ensemble values
27	CRPS	The Continuous Ranked Probability Score
28	CRPSS	The ContinuousRanked Probability Skill Score
29	IGN	The Ignorance Score
30	ME	The Mean Error of the ensemble mean (unperturbed or supplied)
31	RMSE	The Root Mean Square Error of the ensemble mean (unperturbed or supplied)
32	SPREAD	The square root of the mean of the variance of the unperturbed ensemble member values at each observation location
33	ME_OERR	The Mean Error of the PERTURBED ensemble mean (e.g. with Observation Error)
34	RMSE_OERR	The Root Mean Square Error of the PERTURBED ensemble mean (e.g. with Observation Error)
35	SPREAD_OERR	The square root of the mean of the variance of the PERTURBED ensemble member values (e.g. with Observation Error) at each observation location
36	SPREAD_PLUS_OERR	The square root of the sum of unperturbed ensemble variance and the observation error variance

Table 9.3: Format information for RPS (Ranked Probability Score) output line type.

RPS OUTPUT FORMAT		
Column Number	**RPS Column Name**	**Description**
24	RPS	Ranked Probability Score line type
25	TOTAL	Count of observations
26	N_PROB	Number of probability thresholds (i.e. number of ensemble members in Ensemble-Stat)
27	RPS_REL	RPS Reliability, mean of the reliabilities for each RPS threshold
28	RPS_RES	RPS Resolution, mean of the resolutions for each RPS threshold
29	RPS_UNC	RPS Uncertainty, mean of the uncertainties for each RPS threshold
30	RPS	Ranked Probability Score, mean of the Brier Scores for each RPS threshold
31	RPSS	Ranked Probability Skill Score relative to external climatology
32	RPSS_SMPL	Ranked Probability Skill Score relative to sample climatology

Table 9.4: Format information for RHIST (Ranked Histogram) output line type.

RHIST OUTPUT FORMAT		
Column Number	**RHIST Column Name**	**Description**
24	RHIST	Ranked Histogram line type
25	TOTAL	Count of observations
26	N_RANK	Number of possible ranks for observation
27	RANK_i	Count of observations with the i-th rank (repeated)

Table 9.5: Format information for PHIST (Probability Integral Transform Histogram) output line type.

PHIST OUTPUT FORMAT		
Column Number	**PHIST Column Name**	**Description**
24	PHIST	Probability Integral Transform line type
25	TOTAL	Count of observations
26	BIN_SIZE	Probability interval width
27	N_BIN	Total number of probability intervals
28	BIN_i	Count of observations in the ith probability bin (repeated)

Table 9.6: Format information for RELP (Relative Position) output line type.

RELP OUTPUT FORMAT		
Column Number	**RELP Column Name**	**Description**
24	RELP	Relative Position line type
25	TOTAL	Count of observations
26	N_ENS	Number of ensemble members
27	RELP_i	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.

Table 9.7: Format information for ORANK (Observation Rank) output line type.

ORANK OUTPUT FORMAT		
Column Number	**ORANK Column Name**	**Description**
24	ORANK	Observation Rank line type
25	TOTAL	Count of observations
26	INDEX	Line number in ORANK file
27	OBS_SID	Station Identifier
28	OBS_LAT	Latitude of the observation
29	OBS_LON	Longitude of the observation
30	OBS_LVL	Level of the observation
31	OBS_ELV	Elevation of the observation
32	OBS	Value of the observation
33	PIT	Probability Integral Transform
34	RANK	Rank of the observation
35	N_ENS_VLD	Number of valid ensemble values
36	N_ENS	Number of ensemble values
37	ENS_i	Value of the ith ensemble member (repeated)
Last-6	OBS_QC	Quality control string for the observation
Last-5	ENS_MEAN	The unperturbed ensemble mean value
Last-4	CLIMO	The value of the inluded climatology
Last-3	SPREAD	The spread (standard deviation) of the unperturbed ensemble member values
Last-2	ENS_MEAN _OERR	The PERTURBED ensemble mean (e.g. with Observation Error).
Last-1	SPREAD_OERR	The spread (standard deviation) of the PERTURBED ensemble member values (e.g. with Observation Error).
Last	SPREAD_PLUS_OERR	The square root of the sum of the unperturbed ensemble variance and the observation error variance.

Table 9.8: Format information for SSVAR (Spread/Skill Variance) output line type.

SSVAR OUTPUT FORMAT		
Column Number	**SSVAR Column Name**	**Description**
24	SSVAR	Spread/Skill Variance line type
25	TOTAL	Count of observations
26	N_BIN	Number of bins for current forecast run
27	BIN_i	Index of the current bin
28	BIN_N	Number of points in bin i
29	VAR_MIN	Minimum variance
30	VAR_MAX	Maximum variance
31	VAR_MEAN	Average variance
32	FBAR	Average forecast value
33	OBAR	Average observed value
34	FOBAR	Average product of forecast and observation
35	FFBAR	Average of forecast squared
36	OOBAR	Average of observation squared
37-38	FBAR_NCL, FBAR_NCU	Mean forecast normal upper and lower confidence limits
39-41	FSTDEV, FSTDEV_NCL, FSTDEV_NCU	Standard deviation of the error including normal upper and lower confidence limits
42-43	OBAR_NCL, OBAR_NCU	Mean observation normal upper and lower confidence limits
44-46	OSTDEV, OSTDEV_NCL, OSTDEV_NCU	Standard deviation of the error including normal upper and lower confidence limits
47-49	PR_CORR, PR_CORR_NCL, PR_CORR_NCU	Pearson correlation coefficient including normal upper and lower confidence limits
50-52	ME, ME_NCL, ME_NCU	Mean error including normal upper and lower confidence limits
53-55	ESTDEV, ESTDEV_NCL, ESTDEV_NCU	Standard deviation of the error including normal upper and lower confidence limits
56	MBIAS	Magnitude bias
57	MSE	Mean squared error
58	BCMSE	Bias corrected root mean squared error
59	RMSE	Root mean squared error

Chapter 10

Wavelet-Stat Tool

10.1 Introduction

The Wavelet-Stat tool decomposes two-dimensional forecasts and observations according to intensity and scale. This chapter describes the Wavelet-Stat tool, which enables users to apply the Intensity-Scale verification technique described by Casati et al. (2004).

The Intensity-Scale technique is one of the recently developed verification approaches that focus on verification of forecasts defined over spatial domains. Spatial verification approaches, as opposed to point-by-point verification approaches, aim to account for the presence of features and for the coherent spatial structure characterizing meteorological fields. Since these approaches account for the intrinsic spatial correlation existing between nearby grid-points, they do not suffer from point-by-point comparison related verification issues, such as double penalties. Spatial verification approaches aim to account for the observation and forecast time-space uncertainties, and aim to provide feedback on the forecast error in physical terms.

The Intensity-Scale verification technique, as most of the spatial verification approaches, compares a forecast field to an observation field. To apply the Intensity-Scale verification approach, observations need to be defined over the same spatial domain of the forecast to be verified.

Within the spatial verification approaches, the Intensity-Scale technique belongs to the scale-decomposition (or scale-separation) verification approaches. The scale-decomposition approaches enable users to perform the verification on different spatial scales. Weather phenomena on different scales (e.g. frontal systems versus convective showers) are often driven by different physical processes. Verification on different spatial scales can therefore provide deeper insights into model performance at simulating these different processes.

The spatial scale components are obtained usually by applying a single band spatial filter to the forecast and observation fields (e.g. Fourier, Wavelets). The scale-decomposition approaches measure error, bias and skill of the forecast on each different scale component. The scale-decomposition approaches therefore provide feedback on the scale dependency of the error and skill, on the no-skill to skill transition scale, and on the capability of the forecast of reproducing the observed scale structure.

The Intensity-Scale technique evaluates the forecast skill as a function of the intensity values and of the spatial scale of the error. The scale components are obtained by applying a two dimensional Haar wavelet filter. Note that wavelets, because of their locality, are suitable for representing discontinuous fields characterized by few sparse non-zero features, such as precipitation. Moreover, the technique is based on a categorical approach, which is a robust and resistant approach, suitable for non-normally distributed variables, such as precipitation. The intensity-scale technique was specifically designed to cope with the difficult characteristics of precipitation fields, and for the verification of spatial precipitation forecasts. However, the intensity-scale technique can also be applied to verify other variables, such as cloud fraction.

10.2 Scientific and statistical aspects

10.2.1 The method

Casati et al (2004) applied the Intensity-Scale verification to preprocessed and re-calibrated (unbiased) data. The preprocessing was aimed to mainly normalize the data, and defined categorical thresholds so that each categorical bin had a similar sample size. The recalibration was performed to eliminate the forecast bias. Preprocessing and recalibration are not strictly necessary for the application of the Intensity-Scale technique. The MET Intensity-Scale Tool does not perform either, and applies the Intensity-Scale approach to biased forecasts, for categorical thresholds defined by the user.

The Intensity Scale approach can be summarized in the following 5 steps:

1. For each threshold, the forecast and observation fields are transformed into binary fields: where the grid-point precipitation value meets the threshold criteria it is assigned 1, where the threshold criteria are not met it is assigned 0. Figure 10.1 illustrates an example of a forecast and observation fields, and their corresponding binary fields for a threshold of 1mm/h. This case shows an intense storm of the scale of 160 km displaced almost its entire length. The displacement error is clearly visible from the binary field difference and the contingency table image obtained for the same threshold (Table 10.1).

2. The binary forecast and observation fields obtained from the thresholding are then decomposed into the sum of components on different scales, by using a 2D Haar wavelet filter (Figure 10.2). Note that the scale components are fields, and their sum adds up to the original binary field. For a forecast defined over square domain of $\mathbf{2}^{\mathbf{n}}$ x $\mathbf{2}^{\mathbf{n}}$ grid-points, the scale components are $\mathbf{n+1}$: **n** mother wavelet components + the largest father wavelet (or scale-function) component. The **n** mother wavelet components have resolution equal to **1, 2, 4, ...** $\mathbf{2}^{\mathbf{n-1}}$ grid-points. The largest father wavelet component is a constant field over the $\mathbf{2}^{\mathbf{n}}$ x $\mathbf{2}^{\mathbf{n}}$ grid-point domain with value equal to the field mean.

Note that the wavelet transform is a linear operator: this implies that the difference of the spatial scale components of the binary forecast and observation fields (Figure 10.2) are equal to the spatial scale components of the difference of the binary forecast and observation fields (Figure 10.3), and these scale components also add up to the original binary field difference (Figure 10.11). The intensity-scale technique considers

thus the spatial scale of the error. For the case illustrated (Figure 10.1 and Figure 10.3) note the large error associated at the scale of 160 km, due the storm, 160km displaced almost its entire length.

Note also that the means of the binary forecast and observation fields (i.e. their largest father wavelet components) are equal to the proportion of forecast and observed events above the threshold, **(a+b)/n** and **(a+c)/n**, evaluated from the contingency table counts (Table 10.1) obtained from the original forecast and observation fields by thresholding with the same threshold used to obtain the binary forecast and observation fields. This relation is intuitive when observing forecast and observation binary fields and their corresponding contingency table image (Figure 10.1). The comparison of the largest father wavelet component of binary forecast and observation fields therefore provides feedback on the whole field bias.

3. For each threshold (**t**) and for each scale component (**j**) of the binary forecast and observation, the Mean Squared Error (MSE) is then evaluated (Figure 10.4). The error is usually large for small thresholds, and decreases as the threshold increases. This behavior is partially artificial, and occurs because the smaller the threshold the more events will exceed it, and therefore the larger would be the error, since the error tends to be proportional to the amount of events in the binary fields. The artificial effect can be diminished by normalization: because of the wavelet orthogonal properties, the sum of the MSE of the scale components is equal to the MSE of the original binary fields: $\mathrm{MSE}(t) = j\mathrm{MSE}(t,j)$. Therefore, the percentage that the MSE for each scale contributes to the total MSE may be computed: for a given threshold, **t**, $\mathrm{MSE}\%(t,j) = \mathrm{MSE}(t,j)/\mathrm{MSE}(t)$. The MSE% does not exhibit the threshold dependency, and usually shows small errors on large scales and large errors on small scales, with the largest error associated to the smallest scale and highest threshold. For the NIMROD case illustrated, note the large error at 160 km and between the thresholds of and 4 mm/h, due to the storm, 160km displaced almost its entire length.

Note that the MSE of the original binary fields is equal to the proportion of the counts of misses (**c/n**) and false alarms (**b/n**) for the contingency table (Table 10.1) obtained from the original forecast and observation fields by thresholding with the same threshold used to obtain the binary forecast and observation fields: $\mathrm{MSE}(t) = (b+c)/n$. This relation is intuitive when comparing the forecast and observation binary field difference and their corresponding contingency table image (Figure 15.1).

4. The MSE for the random binary forecast and observation fields is estimated by $\mathrm{MSE}(t)\mathrm{random} = \mathrm{FBI} * \mathrm{Br} * (1 - \mathrm{Br}) + \mathrm{Br} * (1 - \mathrm{FBI} * \mathrm{Br})$, where $\mathrm{FBI} = (a+b)/(a+c)$ is the frequency bias index and $\mathrm{Br} = (a+c)/n$ is the sample climatology from the contingency table (Table 10.1) obtained from the original forecast and observation fields by thresholding with the same threshold used to obtain the binary forecast and observation fields. This formula follows by considering the Murphy and Winkler (1987) framework, applying the Bayes' theorem to express the joint probabilities **b/n** and **c/n** as product of the marginal and conditional probability (e.g. Jolliffe and Stephenson, 2003; Wilks, 2006), and then noticing that for a random forecast the conditional probability is equal to the unconditional one, so that **b/n** and **c/n** are equal to the product of the corresponding marginal probabilities solely.

5. For each threshold (**t**) and scale component (**j**), the skill score based on the MSE of binary forecast and observation scale components is evaluated (Figure 10.5). The standard skill score definition as in

Jolliffe and Stephenson (2003) or Wilks (2006) is used, and random chance is used as reference forecast. The MSE for the random binary forecast is equipartitioned on the **n+1** scales to evaluate the skill score: $\mathrm{SS}(t,j) = 1 - \mathrm{MSE}(t,j) * (n+1)/\mathrm{MSE}(t)\mathrm{random}$

The Intensity-Scale (IS) skill score evaluates the forecast skill as a function of the precipitation intensity and of the spatial scale of the error. Positive values of the IS skill score are associated to a skillful forecast, whereas negative values are associated to no skill. Usually large scales exhibit positive skill (large scale events, such as fronts, are well predicted), whereas small scales exhibit negative skill (small scale events, such as convective showers, are less predictable), and the smallest scale and highest thresholds exhibit the worst skill. For the NIMROD case illustrated note the negative skill associated to the 160 km scale, for the thresholds to 4 mm/h, due to the 160 km storm displaced almost its entire length.

Table 10.1: **2x2** contingency table in terms of counts. The $\mathbf{n_{ij}}$ values in the table represent the counts in each forecast-observation category, where **i** represents the forecast and **j** represents the observations.

Forecast	Observation		Total
	o = 1 (e.g., "Yes")	o = 0 (e.g., "No")	
f = 1 (e.g., "Yes")	Hits=**a**	False Alarms = **b**	**a+b**
f = 0 (e.g., "No")	Misses=**c**	Correct rejections = **d**	**c+d**
Total	**a+c**	**b+d**	**a+b+c+d**

In addition to the MSE and the SS, the energy squared is also evaluated, for each threshold and scale (Figure 10.6). The energy squared of a field X is the average of the squared values: $\mathrm{En2}(X) = \sum_i x_i^2$. The energy squared provides feedback on the amount of events present in the forecast and observation fields for each scale, for a given threshold. Usually, small thresholds are associated to a large energy, since many events exceed the threshold. Large thresholds are associated to a small energy, since few events exceed the threshold. Comparison of the forecast and observed squared energy provide feedback on the bias on different scales, for each threshold.

The En2 bias for each threshold and scale is assessed by the En2 relative difference, equal to the difference between forecast and observed squared energies normalized by their sum: $[\mathrm{En2}(F) - \mathrm{En2}(O)]/[\mathrm{En2}(F) + \mathrm{En2}(O)]$. Since defined in such a fashion, the En2 relative difference accounts for the difference between forecast and observation squared energies relative to their magnitude, and it is sensitive therefore to the ratio of the forecast and observed squared energies. The En2 relative difference ranges between -1 and 1, positive values indicate over-forecast and negative values indicate under-forecast. For the NIMROD case illustrated the forecast exhibits over-forecast for small thresholds, quite pronounced on the large scales, and under-forecast for high thresholds.

As for the MSE, the sum of the energy of the scale components is equal to the energy of the original binary field: $\mathrm{En2}(t) = j\mathrm{En2}(t,j)$. Therefore, the percentage that the En2 for each scale contributes the total En2 may be computed: for a given threshold, **t**, $\mathrm{En2\%}(t,j) = \mathrm{En2}(t,j)/\mathrm{En2}(t)$. Usually, for precipitation fields, low thresholds exhibit most of the energy percentage on large scales (and less percentage on the small scales), since low thresholds are associated to large scale features, such as fronts. On the other hand, for higher thresholds the energy percentage is usually larger on small scales, since intense events are associated to small scales features, such as convective cells or showers. The comparison of the forecast and observation

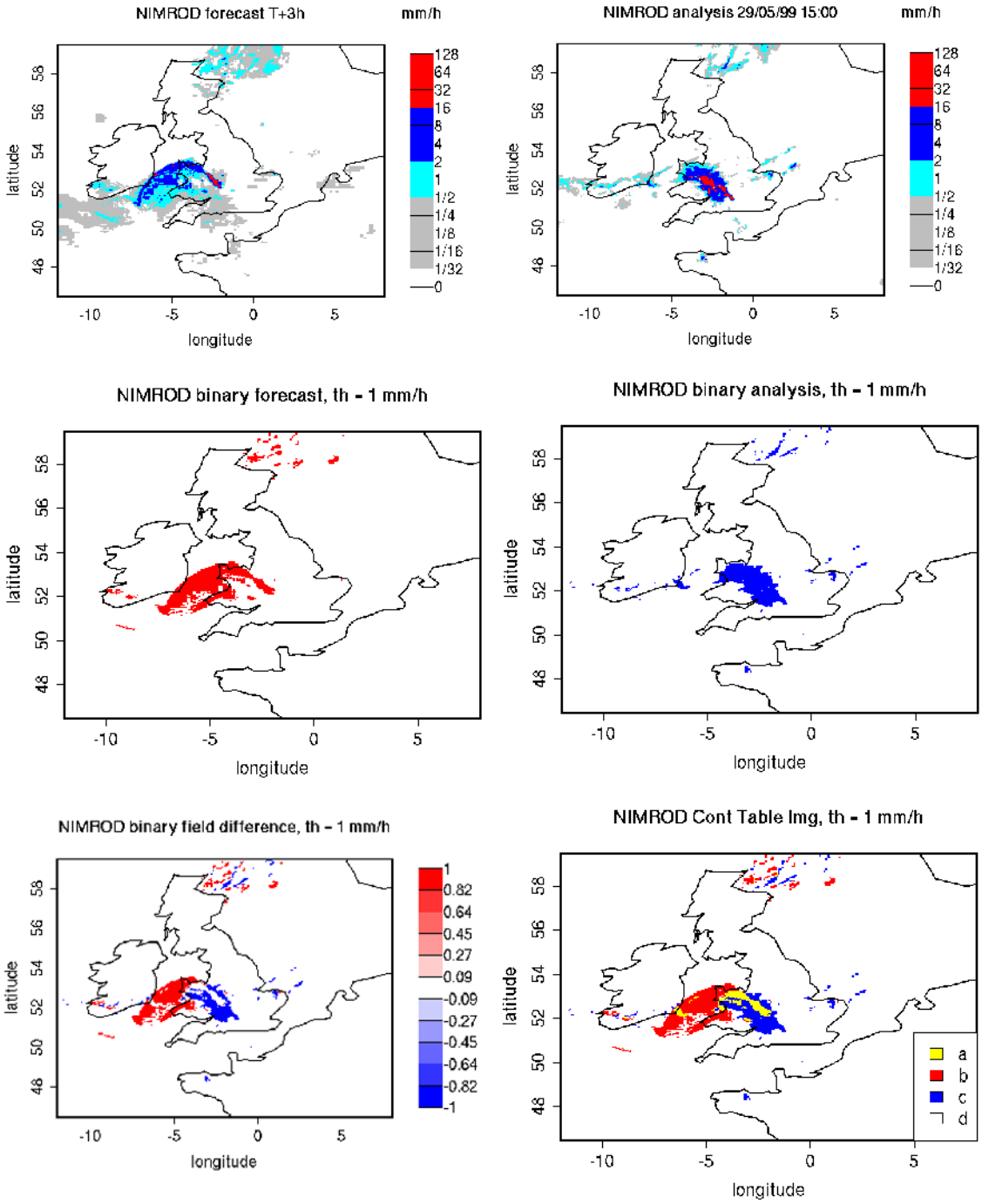

Figure 10.1: NIMROD 3h lead-time forecast and corresponding verifying analysis field (precipitation rate in mm/h, valid the 05/29/99 at 15:00 UTC); forecast and analysis binary fields obtained for a threshold of 1mm/h, the binary field difference has their corresponding Contingency Table Image (see Table 10.1). The forecast shows a storm of 160 km displaced almost its entire length.

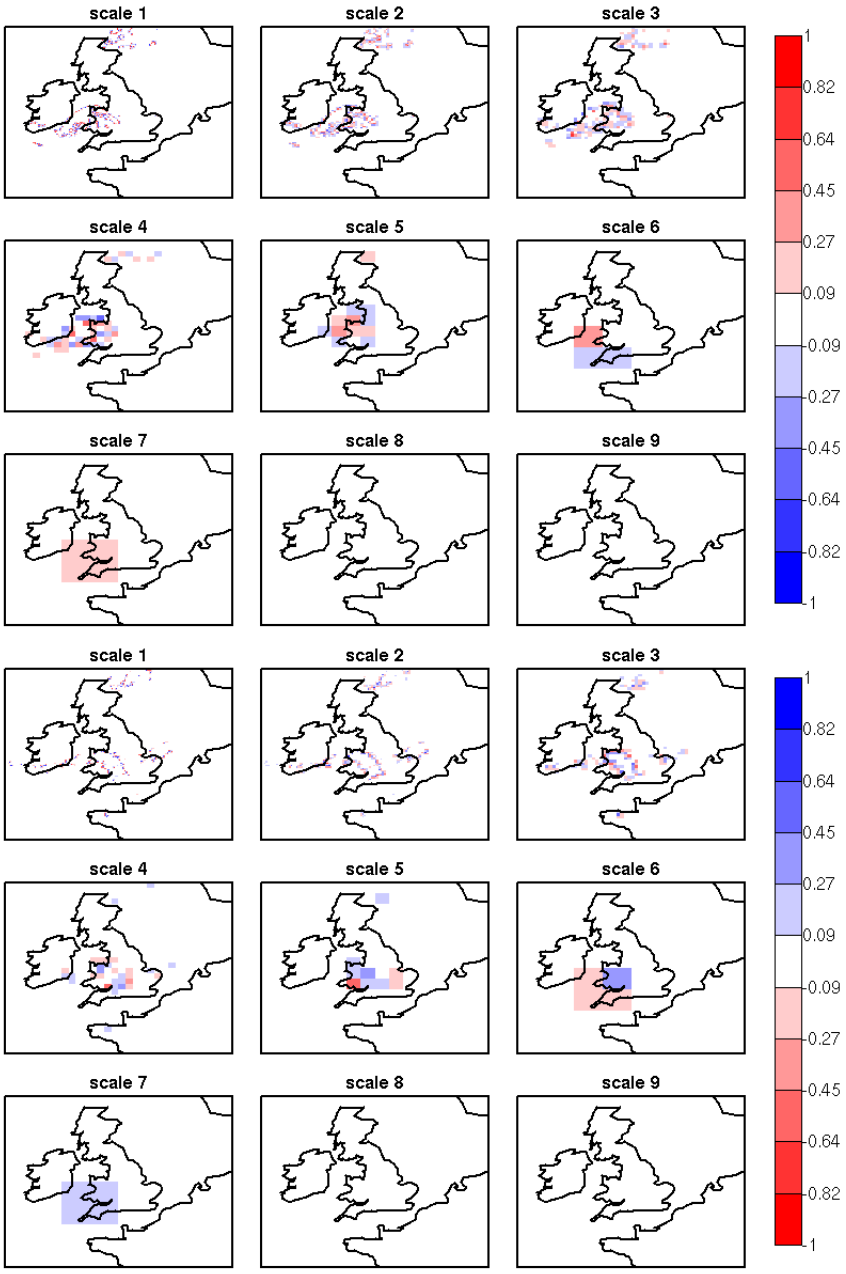

Figure 10.2: NIMROD binary forecast (top) and binary analysis (bottom) spatial scale components obtained by a 2D Haar wavelet transform (th=1 mm/h). Scale 1 to 8 refer to mother wavelet components (5, 10, 20, 40, 80, 160, 320, 640 km resolution); scale 9 refer to the largest father wavelet component (1280 km resolution).

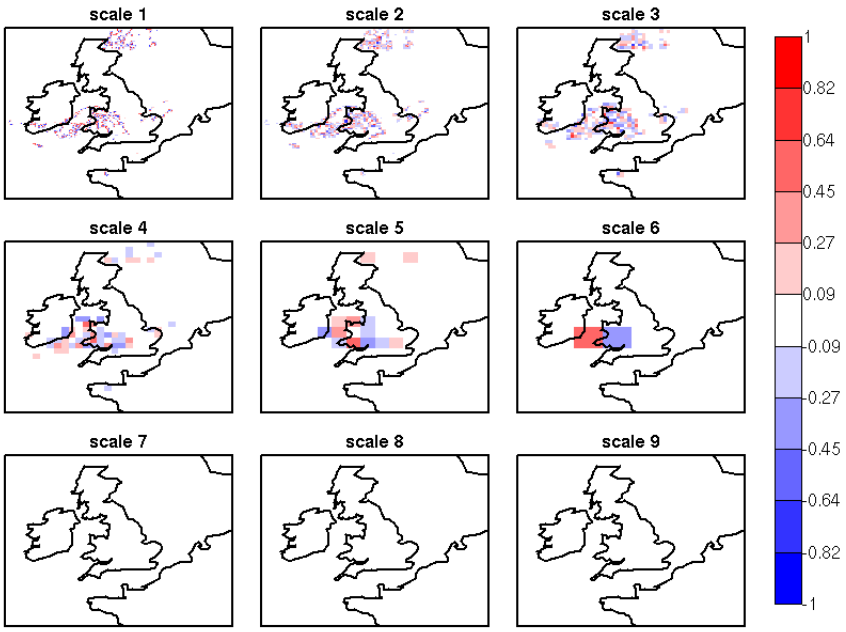

Figure 10.3: NIMROD binary field difference spatial scale components obtained by a 2D Haar wavelet transform (th=1 mm/h). Scales 1 to 8 refer to mother wavelet components (5, 10, 20, 40, 80, 160, 320, 640 km resolution); scale 9 refers to the largest father wavelet component (1280 km resolution). Note the large error at the scale 6 = 160 km, due to the storm, 160 km displaced almost of its entire length.

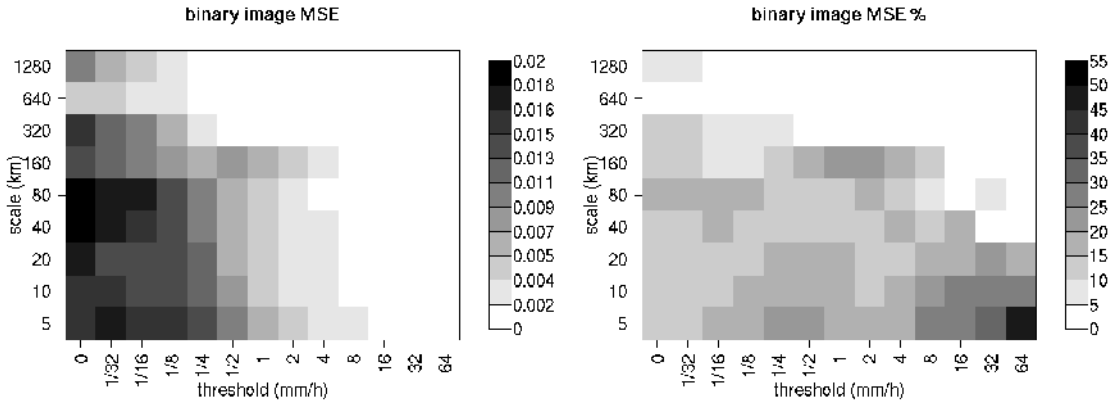

Figure 10.4: MSE and MSE % for the NIMROD binary forecast and analysis spatial scale components. In the MSE%, note the large error associated to the scale 6 = 160 km, for the thresholds $\frac{1}{2}$ to 4 mm/h, associated to the displaced storm.

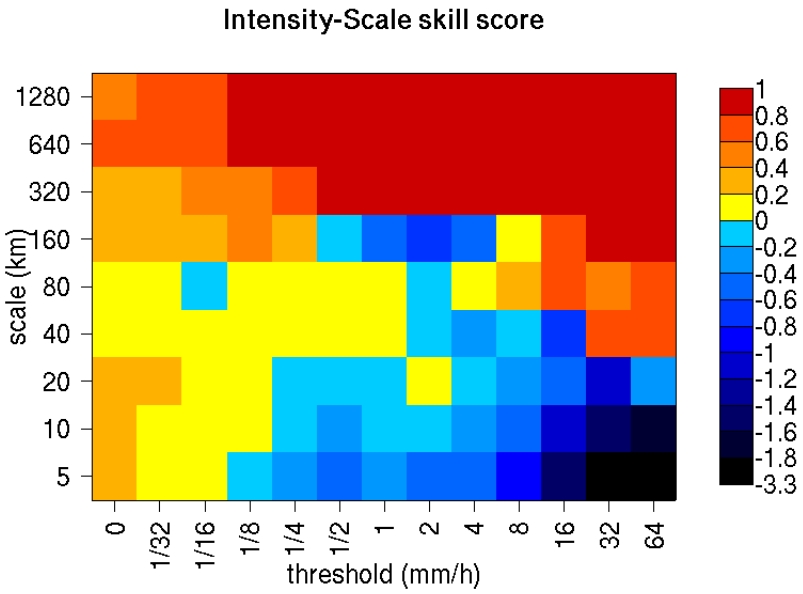

Figure 10.5: Intensity-Scale skill score for the NIMROD forecast and analysis shown in Figure 10.1. The skill score is a function of the intensity of the precipitation rate and spatial scale of the error. Note the negative skill associated to the scale 6 = 160 km, for the thresholds to 4 mm/h, associated to the displaced storm.

squared energy percentages provides feedback on how the events are distributed across the scales, and enable the comparison of forecast and observation scale structure.

For the NIMROD case illustrated, the scale structure is assessed again by the relative difference, but calculated of the squared energy percentages. For small thresholds the forecast over-estimates the number of large scale events and under-estimates the number of small scale events, in proportion to the total number of events. On the other hand, for larger thresholds the forecast under-estimates the number of large scale events and over-estimates the number of small scale events, again in proportion to the total number of events. Overall it appears that the forecast over-estimates the percentage of events associated to high occurrence, and under-estimate the percentage of events associated to low occurrence. The En2% for the 64 mm/h thresholds is homogeneously under-estimated for all the scales, since the forecast does not have any event exceeding this threshold.

Note that the energy squared of the observation binary field is identical to the sample climatology Br = $(a+c)/n$. Similarly, the energy squared of the forecast binary field is equal to $(a+b)/n$. The ratio of the squared energies of the forecast and observation binary fields is equal to the FBI = $(a+b)/(a+c)$FBI=(a+b)/(a+c), for the contingency table (Table 10.1) obtained from the original forecast and observation fields by thresholding with the same threshold used to obtained the binary forecast and observation fields.

10.2.2 The spatial domain constraints

The Intensity-Scale technique is constrained by the fact that orthogonal wavelets (discrete wavelet transforms) are usually performed dyadic domains, square domains of $\mathbf{2}^{\mathrm{n}}$ x $\mathbf{2}^{\mathrm{n}}$ grid-points. The Wavelet-Stat tool

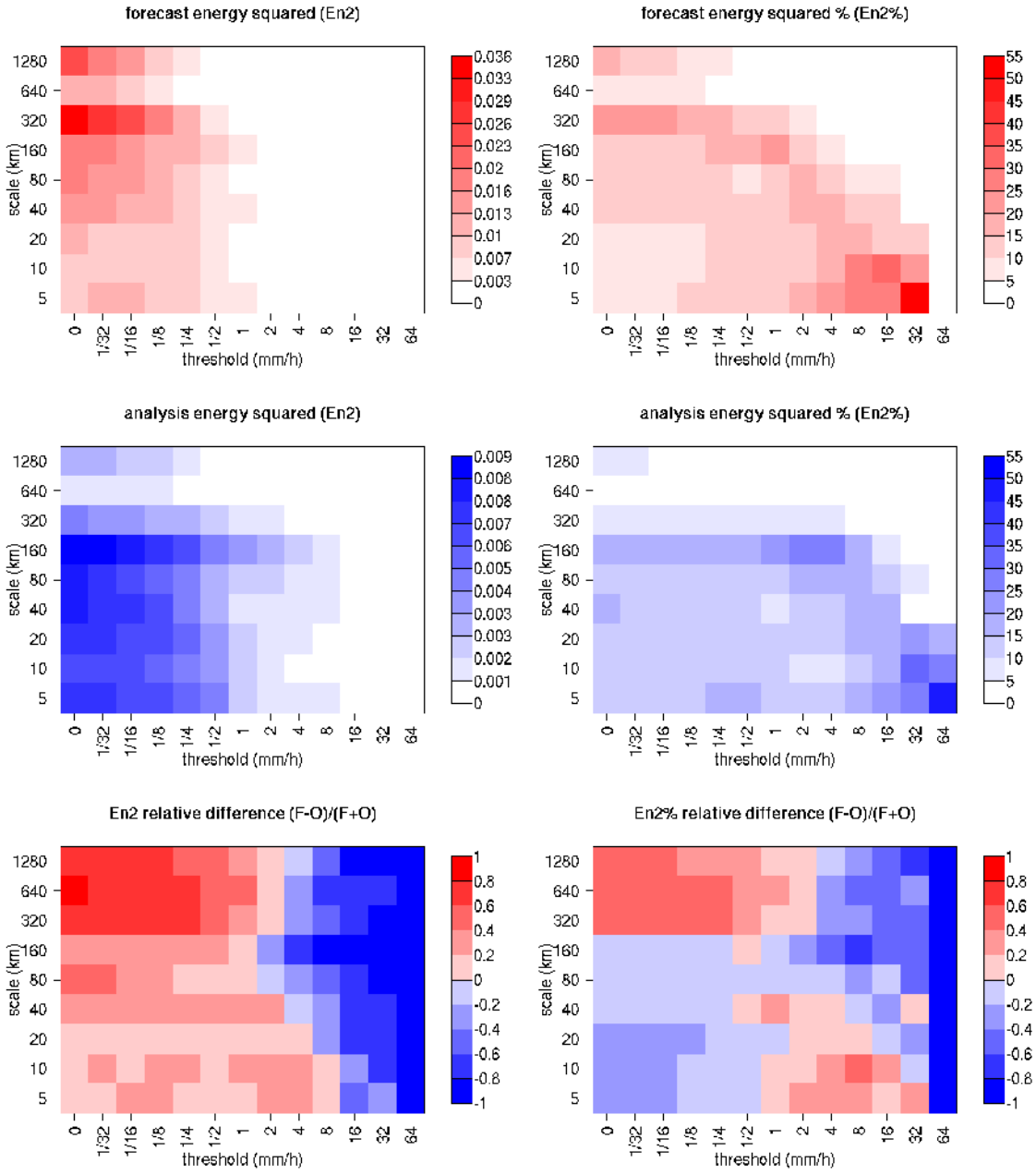

Figure 10.6: Energy squared and energy squared percentages, for each threshold and scale, for the NIMROD forecast and analysis, and forecast and analysis En2 and En2% relative differences.

handles this issue based on settings in the configuration file by defining tiles of dimensions $\mathbf{2}^n$ x $\mathbf{2}^n$ over the input domain in the following ways:

1. User-Defined Tiling: The user may define one or more tiles of size $\mathbf{2}^n$ x $\mathbf{2}^n$ over their domain to be applied. This is done by selecting the grid coordinates for the lower-left corner of the tile(s) and the tile dimension to be used. If the user specifies more than one tile, the Intensity-Scale method will be applied to each tile separately. At the end, the results will automatically be aggregated across all the tiles and written out with the results for each of the individual tiles. Users are encouraged to select tiles which consist entirely of valid data.

2. Automated Tiling: This tiling method is essentially the same as the user-defined tiling method listed above except that the tool automatically selects the location and size of the tile(s) to be applied. It figures out the maximum tile of dimension $\mathbf{2}^n$ x $\mathbf{2}^n$ that fits within the domain and places the tile at the center of the domain. For domains that are very elongated in one direction, it defines as many of these tiles as possible that fit within the domain.

3. Padding: If the domain size is only slightly smaller than $\mathbf{2}^n$ x $\mathbf{2}^n$, for certain variables (e.g. precipitation), it is advisable to expand the domain out to $\mathbf{2}^n$ x $\mathbf{2}^n$ grid-points by adding extra rows and/or columns of fill data. For precipitation variables, a fill value of zero is used. For continuous variables, such as temperature, the fill value is defined as the mean of the valid data in the rest of the field. A drawback to the padding method is the introduction of artificial data into the original field. Padding should only be used when a very small number of rows and/or columns need to be added.

10.2.3 Aggregation of statistics on multiple cases

The Stat-Analysis tool aggregates the intensity scale technique results. Since the results are scale-dependent, it is sensible to aggregate results from multiple model runs (e.g. daily runs for a season) on the same spatial domain, so that the scale components for each singular case will be the same number, and the domain, if not a square domain of $\mathbf{2}^n$ x $\mathbf{2}^n$ grid-points, will be treated in the same fashion. Similarly, the intensity thresholds for each run should all be the same.

The MSE and forecast and observation squared energy for each scale and thresholds are aggregated simply with a weighted average, where weights are proportional to the number of grid-points used in each single run to evaluate the statistics. If the same domain is always used (and it should) the weights result all the same, and the weighted averaging is a simple mean. For each threshold, the aggregated Br is equal to the aggregated squared energy of the binary observation field, and the aggregated FBI is obtained as the ratio of the aggregated squared energies of the forecast and observation binary fields. From aggregated Br and FBI, the MSErandom for the aggregated runs can be evaluated using the same formula as for the single run. Finally, the Intensity-Scale Skill Score is evaluated by using the aggregated statistics within the same formula used for the single case.

10.3 Practical information

The following sections describe the usage statement, required arguments and optional arguments for the Stat-Analysis tool.

10.3.1 wavelet_stat usage

The usage statement for the Wavelet-Stat tool is shown below:

```
Usage: wavelet_stat
       fcst_file
       obs_file
       config_file
       [-outdir path]
       [-log file]
       [-v level]
       [-compress level]
```

wavelet_stat has three required arguments and accepts several optional ones.

Required arguments for wavelet_stat

1. The **fcst_file** argument is the gridded file containing the model data to be verified.
2. The **obs_file** argument is the gridded file containing the observations to be used.
3. The **config_file** argument is the configuration file to be used. The contents of the configuration file are discussed below.

Optional arguments for wavelet_stat

4. The **-outdir path** indicates the directory where output files should be written.
5. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.
6. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity will increase the amount of logging.
7. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

An example of the wavelet_stat calling sequence is listed below:

```
wavelet_stat \
sample_fcst.grb \
sample_obs.grb \
WaveletStatConfig
```

In the example, the Wavelet-Stat tool will verify the model data in the **sample_fcst.grb** GRIB file using the observations in the **sample_obs.grb** GRIB file applying the configuration options specified in the **WaveletStatConfig** file.

10.3.2 wavelet_stat configuration file

The default configuration file for the Wavelet-Stat tool, **WaveletStatConfig_default**, can be found in the installed **share/met/config** directory. Another version of the configuration file is provided in **scripts/-config**. We recommend that users make a copy of the default (or other) configuration file prior to modifying it. The contents are described in more detail below. Note that environment variables may be used when editing configuration files, as described in the Section4.1.2 for the PB2NC tool.

```
model             = "WRF";
desc              = "NA";
obtype            = "ANALYS";
fcst              = { ... }
obs               = { ... }
regrid            = { ... }
mask_missing_flag = NONE;
met_data_dir      = "MET_BASE";
ps_plot_flag      = TRUE;
fcst_raw_plot     = { color_table = "MET_BASE/colortables/met_default.ctable";
                      plot_min = 0.0; plot_max = 0.0; }
obs_raw_plot      = { ... }
wvlt_plot         = { ... }
output_prefix     = "";
version           = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
grid_decomp_flag = AUTO;
tile = {
   width    = 0;
   location = [ { x_ll = 0; y_ll = 0; } ];
}
```

The **grid_decomp_flag** variable specifies how tiling should be performed:

- **AUTO** indicates that the automated-tiling should be done.
- **TILE** indicates that the user-defined tiles should be applied.
- **PAD** indicated that the data should be padded out to the nearest dimension of $\mathbf{2}^n$ x $\mathbf{2}^n$

The **width** and **location** variables allow users to manually define the tiles of dimension they would like to apply. The x_ll and y_ll variables specify the location of one or more lower-left tile grid (x, y) points.

```
wavelet = {
   type   = HAAR;
   member = 2;
}
```

The **wavelet_flag** and **wavelet_k** variables specify the type and shape of the wavelet to be used for the scale decomposition. The Casati et al. (2004) method uses a Haar wavelet which is a good choice for discontinuous fields like precipitation. However, users may choose to apply any wavelet family/shape that is available in the GNU Scientific Library. Values for the **wavelet_flag** variable, and associated choices for k, are described below:

- **HAAR** for the Haar wavelet (member = 2).
- **HAAR_CNTR** for the Centered-Haar wavelet (member = 2).
- **DAUB** for the Daubechies wavelet (member = 4, 6, 8, 10, 12, 14, 16, 18, 20).
- **DAUB_CNTR** for the Centered-Daubechies wavelet (member = 4, 6, 8, 10, 12, 14, 16, 18, 20).
- **BSPLINE** for the Bspline wavelet (member = 103, 105, 202, 204, 206, 208, 301, 303, 305, 307, 309).
- **BSPLINE_CNTR** for the Centered-Bspline wavelet (member = 103, 105, 202, 204, 206, 208, 301, 303, 305, 307, 309).

```
output_flag = {
   isc = BOTH;
}
```

The **output_flag** array controls the type of output that the Wavelet-Stat tool generates. This flags is set similarly to the output flags of the other MET tools, with possible values of NONE, STAT, and BOTH. The ISC line type is the only one available for Intensity-Scale STAT lines.

```
nc_pairs_flag = {
   latlon = TRUE;
   raw    = TRUE;
}
```

The nc_pairs_flag is described in Section 8.3.2

10.3.3 wavelet_stat output

wavelet_stat produces output in STAT and, optionally, ASCII and NetCDF and PostScript formats. The ASCII output duplicates the STAT output but has the data organized by line type. While the Wavelet-Stat tool currently only outputs one STAT line type, additional line types may be added in future releases. The output files are written to the default output directory or the directory specified by the -outdir command line option.

The output STAT file is named using the following naming convention:

wavelet_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV.stat where PREFIX indicates the user-defined output prefix, HHMMSS indicates the forecast lead time, and YYYYMMDD_HHMMSS indicates the forecast valid time.

The output ASCII files are named similarly:

wavelet_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV_TYPE.txt where TYPE is isc to indicate that this is an intensity-scale line type.

The format of the STAT and ASCII output of the Wavelet-Stat tool is similar to the format of the STAT and ASCII output of the Point-Stat tool. Please refer to the tables in Section 7.3.3 for a description of the common output for STAT files types. The information contained in the STAT and isc files are identical. However, for consistency with the STAT files produced by other tools, the STAT file will only have column headers for the first 21 fields. The isc file contains all headers. The format of the ISC line type is explained in the following table.

Table 10.2: Header information for each file wavelet-stat outputs.

HEADER		
Column Number	**Header Column Name**	**Description**
1	VERSION	Version number
2	MODEL	User provided text string designating model name
3	DESC	User provided text string describing the verification task
4	FCST_LEAD	Forecast lead time in HHMMSS format
5	FCST_VALID_BEG	Forecast valid start time in YYYYMMDD_HHMMSS format
6	FCST_VALID_END	Forecast valid end time in YYYYMMDD_HHMMSS format
7	OBS_LEAD	Observation lead time in HHMMSS format
8	OBS_VALID_BEG	Observation valid start time in YYYYMMDD_HHMMSS format
9	OBS_VALID_END	Observation valid end time in YYYYMMDD_HHMMSS format
10	FCST_VAR	Model variable
11	FCST_UNITS	Units for model variable
12	FCST_LEV	Selected Vertical level for forecast
13	OBS_VAR	Observation variable
14	OBS_UNITS	Units for observation variable
15	OBS_LEV	Selected Vertical level for observations
16	OBTYPE	User provided text string designating the observation type
17	VX_MASK	Verifying masking region indicating the masking grid or polyline region applied
18	INTERP_MTHD	NA in Wavelet-Stat
19	INTERP_PNTS	NA in Wavelet-Stat
20	FCST_THRESH	The threshold applied to the forecast
21	OBS_THRESH	The threshold applied to the observations
22	COV_THRESH	NA in Wavelet-Stat
23	ALPHA	NA in Wavelet-Stat
24	LINE_TYPE	See table below.

Table 10.3: Format information for the ISC (Intensity-Scale) output line type.

ISC OUTPUT FORMAT		
Column Number	**ISC Column Name**	**Description**
24	ISC	Intensity-Scale line type
25	TOTAL	The number of grid points (forecast locations) used
26	TILE_DIM	The dimensions of the tile
27	TILE_XLL	Horizontal coordinate of the lower left corner of the tile
28	TILE_YLL	Vertical coordinate of the lower left corner of the tile
29	NSCALE	Total number of scales used in decomposition
30	ISCALE	The scale at which all information following applies
31	MSE	Mean squared error for this scale
32	ISC	The intensity scale skill score
33	FENERGY	Forecast energy squared for this scale
34	OENERGY	Observed energy squared for this scale
35	BASER	The base rate (not scale dependent)
36	FBIAS	The frequency bias

Table 10.4: Dimensions defined in NetCDF output.

wavelet_stat NetCDF DIMENSIONS	
NetCDF Dimension	**Description**
x	Dimension of the tile which equals $\mathbf{2}^{\mathbf{n}}$
y	Dimension of the tile which equals $\mathbf{2}^{\mathbf{n}}$
scale	Dimension for the number of scales. This is set to $\mathbf{n+2}$, where $\mathbf{2}^{\mathbf{n}}$ is the tile dimension. The 2 extra scales are for the binary image and the wavelet averaged over the whole tile.
tile	Dimension for the number of tiles used

Table 10.5: Variables defined in NetCDF output.

wavelet-stat NetCDF VARIABLES		
NetCDF Variable	**Dimension**	**Description**
FCST_FIELD_LEVEL_RAW	tile, x, y	Raw values for the forecast field specified by "FIELD_LEVEL"
OBS_FIELD_LEVEL_RAW	tile, x, y	Raw values for the observation field specified by "FIELD_LEVEL"
DIFF_FIELD_LEVEL_RAW	tile, x, y	Raw values for the difference field (**f-o**) specified by "FIELD_LEVEL"
FCST_FIELD_LEVEL_THRESH	tile, scale, x, y	Wavelet scale-decomposition of the forecast field specified by "FIELD_LEVEL_THRESH"
OBS_FIELD_LEVEL_THRESH	tile, scale, x, y	Wavelet scale-decomposition of the observation field specified by "FIELD_LEVEL_THRESH"

The **Wavelet-Stat** tool creates a NetCDF output file containing the raw and decomposed values for the forecast, observation, and difference fields for each combination of variable and threshold value.

The dimensions and variables included in the wavelet_stat NetCDF files are described in Tables 10.4 and 10.5.

Lastly, the Wavelet-Stat tool creates a PostScript plot summarizing the scale-decomposition approach used in the verification. The PostScript plot is generated using internal libraries and does not depend on an external plotting package. The generation of this PostScript output can be disabled using the **ps_plot_flag** configuration file option.

The PostScript plot begins with one summary page illustrating the tiling method that was applied to the domain. The remaining pages depict the Intensity-Scale method that was applied. For each combination of field, tile, and threshold, the binary difference field (**f-o**) is plotted followed by the difference field for each decomposed scale. Underneath each difference plot, the statistics applicable to that scale are listed. Examples of the PostScript plots can be obtained by running the example cases provided with the MET tarball.

Chapter 11

GSI Tools

Gridpoint Statistical Interpolation (GSI) diagnostic files are binary files written out from the data assimilation code before the first and after each outer loop. The files contain useful information about how a single observation was used in the analysis by providing details such as the innovation (O-B), observation values, observation error, adjusted observation error, and quality control information.

For more detail on generating GSI diagnostic files and their contents, see the GSI User's Guide: `http://www.dtcenter.org/com-GSI/users/docs/index.php`

When MET reads GSI diagnostic files, the innovation (O-B; generated prior to the first outer loop) or analysis increment (O-A; generated after the final outer loop) is split into separate values for the observation (OBS) and the forecast (FCST), where the forecast value corresponds to the background (O-B) or analysis (O-A).

MET includes two tools for processing GSI diagnostic files. The gsid2mpr tool reformats individual GSI diagnostic files into the MET matched pair (MPR) format, similar to the output of the Point-Stat tool. The gsidens2orank tool processes an ensemble of GSI diagnostic files and reformats them into the MET observation rank (ORANK) line type, similar to the output of the Ensemble-Stat tool. The output of both tools may be passed to the Stat-Analysis tool to compute a wide variety of continuous, categorical, and ensemble statistics.

11.1 GSID2MPR tool

This section describes how to run the tool gsid2mpr tool. The gsid2mpr tool reformats one or more GSI diagnostic files into an ASCII matched pair (MPR) format, similar to the MPR output of the Point-Stat tool. The output MPR data may be passed to the Stat-Analysis tool to compute a wide variety of continuous or categorical statistics.

11.1.1 gsid2mpr usage

The usage statement for the gsid2mpr tool is shown below:

```
Usage: gsid2mpr
       gsi_file_1 [gsi_file_2 ... gsi_file_n]
       [-swap]
       [-no_check_dup]
       [-channel n]
       [-set_hdr col_name value]
       [-suffix string]
       [-outdir path]
       [-log file]
       [-v level]
```

gsid2mpr has one required argument and and accepts several optional ones.

Required arguments for gsid2mpr

1. The **gsi_file_1 [gsi_file2 ... gsi_file_n]** argument indicates the GSI diagnostic files (conventional or radiance) to be reformatted.

Optional arguments for gsid2mpr

2. The **-swap** option switches the endianness when reading the input binary files.
3. The **-no_check_dup** option disables the checking for duplicate matched pairs which slows down the tool considerably for large files.
4. The **-channel n** option overrides the default processing of all radiance channels with the values of a comma-separated list.
5. The **-set_hdr col_name value** option specifies what should be written to the output header columns.
6. The **-suffix string** option overrides the default output filename suffix (.stat).
7. The **-outdir path** option overrides the default output directory (./).
8. The **-log file** option outputs log messages to the specified file.
9. The **-v level** option overrides the default level of logging (2).

An example of the gsid2mpr calling sequence is shown below:

```
gsid2mpr diag_conv_ges.mem001 \
-set_hdr MODEL GSI_MEM001 \
-outdir out
```

In this example, the gsid2mpr tool will process a single input file named **diag_conv_ges.mem001** file, set the output **MODEL** header column to **GSI_MEM001**, and write output to the **out** directory. The output file is named the same as the input file but a **.stat** suffix is added to indicate its format.

11.1.2 gsid2mpr output

The gsid2mpr tool performs a simple reformatting step and thus requires no configuration file. It can read both conventional and radiance binary GSI diagnostic files. Support for additional GSI diagnostic file type may be added in future releases. Conventional files are determined by the presence of the string **conv** in the filename. Files that are not conventional are assumed to contain radiance data. Multiple files of either type may be passed in a single call to the gsid2mpr tool. For each input file, an output file will be generated containing the corresponding matched pair data.

The gsid2mpr tool writes the same set of MPR output columns for the conventional and radiance data types. However, it also writes additional columns at the end of the MPR line which depend on the input file type. Those additional columns are described in the following tables.

Table 11.1: Format information for GSI Diagnostic Conventional MPR (Matched Pair) output line type.

GSI DIAGNOSTIC CONVENTIONAL MPR OUTPUT FILE		
Column Number	**Column Name**	**Description**
1-37		Standard MPR columns described in Table 7.21.
38	OBS_PRS	Model pressure value at the observation height (hPa)
39	OBS_ERR_IN	PrepBUFR inverse observation error
40	OBS_ERR_ADJ	read_PrepBUFR inverse observation error
41	OBS_ERR_FIN	Final inverse observation error
42	PREP_USE	read_PrepBUFR usage
43	ANLY_USE	Analysis usage (1 for yes, -1 for no)
44	SETUP_QC	Setup quality control
45	QC_WGHT	Non-linear quality control relative weight

Table 11.2: Format information for GSI Diagnostic Radiance MPR (Matched Pair) output line type.

GSI DIAGNOSTIC RADIANCE MPR OUTPUT FILE		
Column Number	**Column Name**	**Description**
1-37		Standard MPR columns described in Table 7.21.
38	CHAN_USE	Channel used (1 for yes, -1 for no)
39	SCAN_POS	Sensor scan position
40	SAT_ZNTH	Satellite zenith angle (degrees)
41	SAT_AZMTH	Satellite azimuth angle (degrees)
42	SUN_ZNTH	Solar zenith angle (degrees)
43	SUN_AZMTH	Solar azimuth angle (degrees)
44	SUN_GLNT	Sun glint angle (degrees)
45	FRAC_WTR	Fractional coverage by water
46	FRAC_LND	Fractional coverage by land
47	FRAC_ICE	Fractional coverage by ice
48	FRAC_SNW	Fractional coverage by snow
49	SFC_TWTR	Surface temperature over water (K)
50	SFC_TLND	Surface temperature over land (K)
51	SFC_TICE	Surface temperature over ice (K)
52	SFC_TSNW	Surface temperature over snow (K)
53	TSOIL	Soil temperature (K)
54	SOILM	Soil moisture
55	LAND_TYPE	Surface land type
56	FRAC_VEG	Vegetation fraction
57	SNW_DPTH	Snow depth
58	SFC_WIND	Surface wind speed (m/s)
59	FRAC_CLD CLD_LWC	Cloud fraction (%) Cloud liquid water (kg/m**2) (microwave only)
60	CTOP_PRS TC_PWAT	Cloud top pressure (hPa) Total column precip. water (km/m**2) (microwave only)
61	TFND	Foundation temperature: Tr
62	TWARM	Diurnal warming: d(Tw) at depth zob
63	TCOOL	Sub-layer cooling: d(Tc) at depth zob
64	TZFND	d(Tz)/d(Tr)
65	OBS_ERR	Inverse observation error
66	FCST_NOBC	Brightness temperature with no bias correction (K)
67	SFC_EMIS	Surface emissivity
68	STABILITY	Stability index
69	PRS_MAX_WGT	Pressure of the maximum weighing function

The gsid2mpr output may be passed to the Stat-Analysis tool to derive additional statistics. In particular, users should consider running the **aggregate_stat** job type to read MPR lines and compute partial sums (SL1L2), continuous statistics (CNT), contingency table counts (CTC), or contingency table statistics (CTS). Stat-Analysis has been enhanced to parse any extra columns found at the end of the input lines. Users can filter the values in those extra columns using the **-column_thresh** and **-column_str** job command options.

An example of the Stat-Analysis calling sequence is shown below:

```
stat_analysis -lookin diag_conv_ges.mem001.stat \
```

```
-job aggregate_stat -line_type MPR -out_line_type CNT \
-fcst_var t -column_thresh ANLY_USE eq1
```

In this example, the Stat-Analysis tool will read MPR lines from the input file named **diag_conv_ges.mem001.stat**, retain only those lines where the **FCST_VAR** column indicates temperature (**t**) and where the **ANLY_USE** column has a value of 1.0, and derive continuous statistics.

11.2 GSIDENS2ORANK tool

This section describes how to run the tool gsidens2orank tool. The gsidens2orank tool processes an ensemble of GSI diagnostic files and reformats them into the MET observation rank (ORANK) line type, similar to the output of the Ensemble-Stat tool. The ORANK line type contains ensemble matched pair information and is analogous to the MPR line type for a deterministic model. The output ORANK data may be passed to the Stat-Analysis tool to compute ensemble statistics.

11.2.1 gsidens2orank usage

The usage statement for the gsidens2orank tool is shown below:

```
Usage: gsidens2orank
       ens_file_1 ... ens_file_n | ens_file_list
       -out path
       [-ens_mean path]
       [-swap]
       [-rng_name str]
       [-rng_seed str]
       [-set_hdr col_name value]
       [-log file]
       [-v level]
```

gsidens2orank has three required arguments and accept several optional ones.

Required arguments for gsidens2orank

1. The **ens_file_1 ... ens_file_n** argument is a list of ensemble binary GSI diagnostic files to be reformatted.

2. The **ens_file_list** argument is an ASCII file containing a list of ensemble GSI diagnostic files.

3. The **-out path** argument specifies the name of the output **.stat** file.

Optional arguments for gsidens2orank

4. The **-ens_mean path** option is the ensemble mean binary GSI diagnostic file.
5. The **-swap** option switches the endianness when reading the input binary files.
6. The **-channel n** option overrides the default processing of all radiance channels with a comma-separated list.
7. The **-rng_name str** option overrides the default random number generator name (mt19937).
8. The **-rng_seed str** option overrides the default random number generator seed.
9. The **-set_hdr col_name value** option specifies what should be written to the output header columns.
10. The **-log file** option outputs log messages to the specified file.
11. The **-v level** option overrides the default level of logging (2).

An example of the gsidens2orank calling sequence is shown below:

```
gsidens2orank diag_conv_ges.mem* \
-ens_mean diag_conv_ges.ensmean \
-out diag_conv_ges_ens_mean_orank.txt
```

In this example, the gsidens2orank tool will process all of the ensemble members whose file name matches **diag_conv_ges.mem***, write output to the file named **diag_conv_ges_ens_mean_orank.txt**, and populate the output **ENS_MEAN** column with the values found in the **diag_conv_ges.ensmean** file rather than computing the ensemble mean values from the ensemble members on the fly.

11.2.2 gsidens2orank output

The gsidens2orank tool performs a simple reformatting step and thus requires no configuration file. The multiple files passed to it are interpreted as members of the same ensemble. Therefore, each call to the tool processes exactly one ensemble. All input ensemble GSI diagnostic files must be of the same type. Mixing conventional and radiance files together will result in a runtime error. The gsidens2orank tool processes each ensemble member and keeps track of the observations it encounters. It constructs a list of the ensemble values corresponding to each observation and writes an output ORANK line listing the observation value, its rank, and all the ensemble values. The random number generator is used by the gsidens2orank tool to randomly assign a rank value in the case of ties.

The gsid2mpr tool writes the same set of ORANK output columns for the conventional and radiance data types. However, it also writes additional columns at the end of the ORANK line which depend on the input file type. The extra columns are limited to quantities which remain constant over all the ensemble members

and are therefore largely a subset of the extra columns written by the gsid2mpr tool. Those additional columns are described in the following tables.

Table 11.3: Format information for GSI Diagnostic Conventional ORANK (Observation Rank) output line type.

GSI DIAGNOSTIC CONVENTIONAL ORANK OUTPUT FILE		
Column Number	**Column Name**	**Description**
1-?		Standard ORANK columns described in Table 9.7.
Last-2	N_USE	Number of members with ANLY_USE = 1
Last-1	PREP_USE	read_PrepBUFR usage
Last	SETUP_QC	Setup quality control

Table 11.4: Format information for GSI Diagnostic Radiance ORANK (Observation Rank) output line type.

GSI DIAGNOSTIC RADIANCE ORANK OUTPUT FILE		
Column Number	**Column Name**	**Description**
1-?		Standard ORANK columns described in Table 9.7.
Last-24	N_USE	Number of members with OBS_QC = 0
Last-23	CHAN_USE	Channel used (1 for yes, -1 for no)
Last-22	SCAN_POS	Sensor scan position
Last-21	SAT_ZNTH	Satellite zenith angle (degrees)
Last-20	SAT_AZMTH	Satellite azimuth angle (degrees)
Last-19	SUN_ZNTH	Solar zenith angle (degrees)
Last-18	SUN_AZMTH	Solar azimuth angle (degrees)
Last-17	SUN_GLNT	Sun glint angle (degrees)
Last-16	FRAC_WTR	Fractional coverage by water
Last-15	FRAC_LND	Fractional coverage by land
Last-14	FRAC_ICE	Fractional coverage by ice
Last-13	FRAC_SNW	Fractional coverage by snow
Last-12	SFC_TWTR	Surface temperature over water (K)
Last-11	SFC_TLND	Surface temperature over land (K)
Last-10	SFC_TICE	Surface temperature over ice (K)
Last-9	SFC_TSNW	Surface temperature over snow (K)
Last-8	TSOIL	Soil temperature (K)
Last-7	SOILM	Soil moisture
Last-6	LAND_TYPE	Surface land type
Last-5	FRAC_VEG	Vegetation fraction
Last-4	SNW_DPTH	Snow depth
Last-3	TFND	Foundation temperature: Tr
Last-2	TWARM	Diurnal warming: d(Tw) at depth zob
Last-1	TCOOL	Sub-layer cooling: d(Tc) at depth zob
Last	TZFND	d(Tz)/d(Tr)

The gsidens2orank output may be passed to the Stat-Analysis tool to derive additional statistics. In particular, users should consider running the **aggregate_stat** job type to read ORANK lines and ranked histograms (RHIST), probability integral transform histograms (PHIST), and spread-skill variance output (SSVAR). Stat-Analysis has been enhanced to parse any extra columns found at the end of the input lines.

Users can filter the values in those extra columns using the **-column_thresh** and **-column_str** job command options.

An example of the Stat-Analysis calling sequence is shown below:

```
stat_analysis -lookin diag_conv_ges_ens_mean_orank.txt \
-job aggregate_stat -line_type ORANK -out_line_type RHIST \
-by fcst_var -column_thresh N_USE eq20
```

In this example, the Stat-Analysis tool will read ORANK lines from **diag_conv_ges_ens_mean_orank.txt**, retain only those lines where the **N_USE** column indicates that all 20 ensemble members were used, and write ranked histogram (RHIST) output lines for each unique value of encountered in the **FCST_VAR** column.

Chapter 12

Stat-Analysis Tool

12.1 Introduction

The Stat-Analysis tool ties together results from the Point-Stat, Grid-Stat, Ensemble-Stat, Wavelet-Stat, and TC-Gen tools by providing summary statistical information and a way to filter their STAT output files. It processes the STAT output created by the other MET tools in a variety of ways which are described in this chapter.

MET version 9.0 adds support for the passing matched pair data (MPR) into Stat-Analysis using a Python script with the "-lookin python ..." option. An example of running Stat-Analysis with Python embedding is shown in Section 12.3.1.

12.2 Scientific and statistical aspects

The Stat-Analysis tool can perform a variety of analyses, and each type of analysis is called a "job". The job types include the ability to (i) aggregate results over a user-specified time; (ii) stratify statistics based on time of day, model initialization time, lead-time, model run identifier, output filename, or wavelet decomposition scale; and (iii) compute specific verification indices such as the GO Index[1] and wind direction statistics. Future functionality may include information about time-trends and/or calculations based on climatology (e.g., anomaly correlation). This section summarizes the capabilities of the supported Stat-Analysis jobs.

12.2.1 Filter STAT lines

The Stat-Analysis "filter" job simply filters out specific STAT lines based on user-specified search criteria. All of the STAT lines that are retained from one or many files are written to a single output file. The output file for filtered STAT lines must be specified using the **-dump_row** job command option.

[1] The GO Index is a summary measure for NWP models that is used by the US Air Force. It combines verification statistics for several forecast variables and lead times.

12.2.2 Summary statistics for columns

The Stat-Analysis "summary" job produces summary information for columns of data. After the user specifies the column(s) of interest and any other relevant search criteria, summary information is produced from values in those column(s) of data. The summary statistics produced are: mean, standard deviation, minimum, maximum, the 10th, 25th, 50th, 75th, and 90th percentiles, the interquartile range, the range, and both weighted and unweighted means using the logic prescribed by the World Meteorological Organization (WMO).

Confidence intervals are computed for the mean and standard deviation of the column of data. For the mean, the confidence interval is computed two ways - based on an assumption of normality and also using the bootstrap method. For the standard deviation, the confidence interval is computed using the bootstrap method. In this application of the bootstrap method, the values in the column of data being summarized are resampled, and for each replicated sample, the mean and standard deviation are computed.

The columns to be summarized can be specified in one of two ways. Use the **-line_type** option exactly once to specify a single input line type and use the **-column** option one or more times to select the columns of data to be summarized. Alternatively, use the **-column** option one or more times formatting the entries as **LINE_TYPE:COLUMN**. For example, the RMSE column from the CNT line type can be selected using **-line_type CNT -column RMSE** or using **-column CNT:RMSE**. With the second option, columns from multiple input line types may be selected. For example, **-column CNT:RMSE,CNT:MAE,CTS:CSI** select two CNT columns one CTS column.

The WMO mean values are computed in one of three ways, as determined by the configuration file settings for **wmo_sqrt_stats** and **wmo_fisher_stats**. The statistics listed in the first option are square roots. When computing WMO means, the input values are first squared, then averaged, and the square root of the average value is reported. The statistics listed in the second option are correlations to which the Fisher transformation is applied. For any statistic not listed, the WMO mean is computed as a simple arithmetic mean. The **WMO_TYPE** output column indicates the method applied (**SQRT**, **FISHER**, or **MEAN**). The **WMO_MEAN** and **WMO_WEIGHTED_MEAN** columns contain the unweighted and weighted means, respectively. The value listed in the **TOTAL** column of each input line is used as the weight.

The **-derive** job command option can be used to perform the derivation of statistics on the fly from input partial sums and contingency table counts. When enabled, SL1L2 and SAL1L2 input lines are converted to CNT statistics, VL1L2 input lines are converted to VCNT statistics, and CTC lines are converted to CTS statistics. Users should take care with this option. If the data passed to this job contains both partial sums and derived statistics, using the **-derive** option will effectively cause the statistics to be double counted. Use the **-line_type** job command option to filter the data passed to Stat-Analysis jobs.

12.2.3 Aggregated values from multiple STAT lines

The Stat-Analysis "aggregate" job aggregates values from multiple STAT lines of the same type. The user may specify the specific line type of interest and any other relevant search criteria. The Stat-Analysis tool

then creates sums of each of the values in all lines matching the search criteria. The aggregated data are output as the same line type as the user specified. The STAT line types which may be aggregated in this way are the contingency table (FHO, CTC, PCT, MCTC, NBRCTC), partial sums (SL1L2, SAL1L2, VL1L2, and VAL1L2), and other (ISC, ECNT, RPS, RHIST, PHIST, RELP, NBRCNT, SSVAR, and GRAD) line types.

12.2.4 Aggregate STAT lines and produce aggregated statistics

The Stat-Analysis "aggregate-stat" job aggregates multiple STAT lines of the same type together and produces relevant statistics from the aggregated line. This may be done in the same manner listed above in 12.2.3. However, rather than writing out the aggregated STAT line itself, the relevant statistics generated from that aggregated line are provided in the output. Specifically, if a contingency table line type (FHO, CTC, PCT, MCTC, or NBRCTC) has been aggregated, a contingency table statistics (CTS, PSTD, MCTS, or NBRCTS) line type will be written out. If a partial sums line type (SL1L2 or SAL1L2) has been aggregated, a continuous statistics (CNT) line type will be written out. If a vector partial sums line type (VL1L2) has been aggregated, the vector continuous statistics (VCNT) line type will be written out. For ensembles, the ORANK line type can be accumulated into ECNT, RPS, RHIST, PHIST, RELP, or SSVAR output. If the matched pair line type (MPR) has been aggregated, the user may choose the line type to be output (FHO, CTC, CTS, CNT, MCTC, MCTS, SL1L2, SAL1L2, VL1L2, VCNT, WDIR, PCT, PSTD, PJC, PRC, or ECLV).

When aggregating the matched pair line type (MPR) and computing an output contingency table statistics (CTS) or continuous statistics (CNT) line type, the bootstrapping method is applied for computing confidence intervals. The bootstrapping method is applied here in the same way that it is applied in the statistics tools. For a set of n matched forecast-observation pairs, the matched pairs are resampled with replacement many times. For each replicated sample, the corresponding statistics are computed. The confidence intervals are derived from the statistics computed for each replicated sample.

12.2.5 Skill Score Index, including GO Index

The Stat-Analysis "ss_index" and "go_index" jobs calculate the skill score indices by weighting scores for different meteorological fields at different pressure levels and for different lead times. The GO Index is a special case of the Skill Score index for which a specific configuration file is provided. The GO index is a weighted average of the RMSE values for wind speed, dew point temperature, temperature, height, and pressure at several levels in the atmosphere. The variables, levels, and lead times included in the index are shown in Table 12.1 and are defined by a default Stat-Analysis configuration file. The partial sums (SL1L2 lines in the STAT output) for each of these variables at each level and lead time must have been computed in a previous step. The Stat-Analysis tool then uses the weights in Table 12.1 to compute values for the GO Index. For a general skill score index, the user can specify the weights and variables to use in the calculations in a Stat-Analysis configuration file and run the ss_index job type.

Table 12.1: Variables, levels, and weights used to compute the GO Index.

Variable	Level	Weights by Lead time			
		12 h	24 h	36 h	48 h
Wind speed	250 hPa	4	3	2	1
	400 hPa	4	3	2	1
	850 hPa	4	3	2	1
	Surface	8	6	4	2
Dew point temperature	400 hPa	8	6	4	2
	700 hPa	8	6	4	2
	850 hPa	8	6	4	2
	Surface	8	6	4	2
Temperature	400 hPa	4	3	2	1
	Surface	8	6	4	2
Height	400 hPa	4	3	2	1
Pressure	Mean sea level	8	6	4	2

12.2.6 Ramp Events

The Stat-Analysis "ramp" job identifies ramp events (large increases or decreases in values over a time window) in both the forecast and observation data. It categorizes these events as hits, misses, false alarms, or correct negatives by applying a configurable matching time window and computes the corresponding categorical statistics.

12.2.7 Wind Direction Statistics

The Stat-Analysis "aggregate_stat" job can read vector partial sums and derive wind direction error statistics (WDIR). The vector partial sums (VL1L2 or VAL1L2) or matched pairs (MPR) for the UGRD and VGRD must have been computed in a previous step, i.e. by Point-Stat or Grid-Stat tools. This job computes an average forecast wind direction and an average observed wind direction along with their difference. The output is in degrees. In Point-Stat and Grid-Stat, the UGRD and VGRD can be verified using thresholds on their values or on the calculated wind speed. If thresholds have been applied, the wind direction statistics are calculated for each threshold.

The first step in verifying wind direction is running the Grid-Stat and/or Point-Stat tools to verify each forecast of interest and generate the VL1L2 or MPR line(s). When running these tools, please note:

1. To generate VL1L2 or MPR lines, the user must request the verification of both the U-component and V-component of wind at the same vertical levels.
2. To generate VL1L2 or MPR lines, the user must set the "output_flag" to indicate that the VL1L2 or MPR line should be computed and written out.
3. The user may select one or more spatial verification regions over which to accumulate the statistics.

4. The user may select one or more wind speed thresholds to be applied to the U and V wind components when computing the VL1L2 lines. It may be useful to investigate the performance of wind forecasts using multiple wind speed thresholds. For MPR line types, the wind speed threshold can be applied when computing the MPR lines, or the MPR output may be filtered afterwards by the Stat-Analysis tool.

Once the appropriate lines have been generated for each verification time of interest, the user may run the Stat-Analysis tool to analyze them. The Stat-Analysis job "aggregate_stat", along with the "-output_line_type WDIR" option, reads all of the input lines and computes statistics about the wind direction. When running this job the user is encouraged to use the many Stat-Analysis options to filter the input lines down to the set of lines of interest. The output of the wind direction analysis job consists of two lines with wind direction statistics computed in two slightly different ways. The two output lines begin with "ROW_MEAN_WDIR" and "AGGR_WDIR", and the computations are described below:

1. For the "ROW_MEAN_WDIR" line, each of the input VL1L2 lines is treated separately and given equal weight. The mean forecast wind direction, mean observation wind direction, and the associated error are computed for each of these lines. Then the means are computed across all of these forecast wind directions, observation wind directions, and their errors.

2. For the "AGGR_WDIR" line, the input VL1L2 lines are first aggregated into a single line of partial sums where the weight for each line is determined by the number of points it represents. From this aggregated line, the mean forecast wind direction, observation wind direction, and the associated error are computed and written out.

12.3 Practical information

The following sections describe the usage statement, required arguments and optional arguments for the Stat-Analysis tool.

12.3.1 stat_analysis usage

The usage statement for the Stat-Analysis tool is shown below:

```
Usage: stat_analysis
       -lookin path
       [-out file]
       [-tmp_dir path]
       [-log file]
       [-v level]
       [-config config_file] | [JOB COMMAND LINE]
```

stat_analysis has two required arguments and accepts several optional ones.

In the usage statement for the Stat-Analysis tool, some additional terminology is introduced. In the Stat-Analysis tool, the term "job" refers to a set of tasks to be performed after applying user-specified options (i.e., "filters"). The filters are used to pare down a collection of output from the MET statistics tools to only those lines that are desired for the analysis. The job and its filters together comprise the "job command line". The "job command line" may be specified either on the command line to run a single analysis job or within the configuration file to run multiple analysis jobs at the same time. If jobs are specified in both the configuration file and the command line, only the jobs indicated in the configuration file will be run. The various jobs types are described in Table 10.3 and the filtering options are described in Section10.3.2.

Required arguments for stat_analysis

1. The **-lookin path** specifies the name of a directory to be searched recursively for STAT files (ending in ".stat") or any explicit file name with any suffix (such as "_ctc.txt") to be read. This option may be used multiple times to specify multiple directories and/or files to be read. If "-lookin python" is used, it must be followed a Python embedding script and any command line arguments it takes. Python embedding can be used to pass matched pair (MPR) lines as input to Stat-Analysis.

2. Either a configuration file must be specified with the **-config** option, or a **JOB COMMAND LINE** must be denoted. The **JOB COMMAND LINE** is described in Section12.3.2

Optional arguments for stat_analysis

3. The **-config config_file** specifies the configuration file to be used. The contents of the configuration file are discussed below.

4. The **-out file** option indicates the file to which output data should be written. If this option is not used, the output is directed to standard output.

5. The **-tmp_dir path** option selects the directory for writing out temporary files.

6. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

7. The **-v level** indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity will increase the amount of logging.

An example of the **stat_analysis** calling sequence is shown below.

```
stat_analysis -lookin ../out/point_stat \
-config STATAnalysisConfig
```

In this example, the Stat-Analysis tool will search for valid STAT lines located in the **../out/point_stat** directory that meet the options specified in the configuration file, **config/STATAnalysisConfig**.

12.3.1.1 Python Embedding for Matched Pairs

The example below uses Python embedding.

```
stat_analysis \
-lookin python MET_BASE/python/read_ascii_mpr.py point_stat_mpr.txt \
-job aggregate_stat -line_type MPR -out_line_type CNT \
-by FCST_VAR,FCST_LEV
```

In this example, rather than passing the MPR output lines from Point-Stat directly into Stat-Analysis (which is the typical approach), the read_ascii_mpr.py Python embedding script reads that file and passes the data to Stat-Analysis. The aggregate_stat job is defined on the command line and CNT statistics are derived from the MPR input data. Separate CNT statistics are computed for each unique combination of FCST_VAR and FCST_LEV present in the input. Please refer to Appendix F for more details about Python embedding in MET.

12.3.2 stat_analysis configuration file

The default configuration file for the Stat-Analysis tool named **STATAnalysisConfig_default** can be found in the installed **share/met/config** directory. The version used for the example run in Chapter 2 is also available in **scripts/config**. Like the other configuration files described in this document, it is recommended that users make a copy of these files prior to modifying their contents.

The configuration file for the Stat-Analysis tool is optional. Users may find it more convenient initially to run Stat-Analysis jobs on the command line specifying job command options directly. Once the user has a set of or more jobs they would like to run routinely on the output of the MET statistics tools, they may find grouping those jobs together into a configuration file to be more convenient.

Most of the user-specified parameters listed in the Stat-Analysis configuration file are used to filter the ASCII statistical output from the MET statistics tools down to a desired subset of lines over which statistics are to be computed. Only output that meet all of the parameters specified in the Stat-Analysis configuration file will be retained.

The Stat-Analysis tool actually performs a two step process when reading input data. First, it stores the filtering information defined top section of the configuration file. It applies that filtering criteria when reading the input STAT data and writes the filtered data out to a temporary file. Second, each job defined in the **jobs** entry reads data from that temporary file and performs the task defined for the job. After all jobs have run, the Stat-Analysis tool deletes the temporary file.

This two step process enables the Stat-Analysis tool to run more efficiently when many jobs are defined in the configuration file. If only operating on a small subset of the input data, the common filtering criteria can be applied once rather than re-applying it for each job. In general, filtering criteria common to all tasks defined in the **jobs** entry should be moved to the top section of the configuration file.

As described above, filtering options specified in the first section of the configuration file will be applied to every task in the **jobs** entry. However, if an individual job specifies a particular option that was specified above, it will be applied for that job. For example, if the **model**[] option is set at the top to ["Run 1", "Run2"], but a job in the joblist sets the **-model** option as "Run1", that job will be performed only on "Run1" data. Also note that environment variables may be used when editing configuration files, as described in the Section4.1.2 for the PB2NC tool.

```
boot           = { interval = PCTILE; rep_prop = 1.0; n_rep = 1000;
                   rng = "mt19937"; seed = ""; }
rank_corr_flag = TRUE;
tmp_dir        = "/tmp";
version        = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
model = [];
```

The user may specify a comma-separated list of model names to be used for all analyses performed. The names must be in double quotation marks. If multiple models are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-model**" option within the job command lines.

```
desc = [];
```

The user may specify a comma-separated list of description strings to be used for all analyses performed. The names must be in double quotation marks. If multiple description strings are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-desc**" option within the job command lines.

```
fcst_lead = [];
obs_lead  = [];
```

The user may specify a comma-separated list of forecast and observation lead times in HH[MMSS] format to be used for any analyses to be performed. If multiple times are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-fcst_lead**" and "**-obs_lead**" options within the job command lines.

```
fcst_valid_beg  = "";
fcst_valid_end  = "";
fcst_valid_hour = "";
obs_valid_beg   = "";
obs_valid_end   = ""
obs_valid_hour  = "";
```

The user may specify the beginning, ending, and instantaneous valid times in YYYYMMDD[_HH[MMSS]] format to be used for all analyses performed. If multiple valid times fall within the valid time window, the analyses will be performed on their union. These selections may be further refined by using the "**-fcst_valid_beg**", "**-fcst_valid_end**", "**-obs_valid_beg**", "**-obs_valid_end**", "**fcst_valid_hour**" and "**-obs_valid_hour**" options within the job command line.

```
fcst_init_beg  = "";
fcst_init_end  = "";
fcst_init_hour = "";
obs_init_beg   = "";
obs_init_end   = "";
obs_init_hour  = "";
```

The user may specify the beginning, ending, or exact model initialization times in YYYYMMDD[_HH[MMSS]] format to be used for all analyses performed. If multiple init times fall within the init time window, the analyses will be performed on their union. These selections may be further refined by using the **"-fcst_init_beg", "-fcst_init_end", "-obs_init_beg", "-obs_init_end", fcst_init_hour"** and **"-obs_init_hour"** options within the job command line.

```
fcst_var = [];
obs_var  = [];
```

The user may specify a comma-separated list of forecast and observation variable types to be used for any analyses to be performed. If multiple variable types are listed, the analyses will be performed on their union.

These selections may be further refined by using the "**-fcst_var**" and "**-obs_var**" options within the job command lines.

```
fcst_units = [];
obs_units  = [];
```

The user may specify a comma-separated list of forecast and observation units to be used for any analyses to be performed. If multiple units are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-fcst_units**" and "**-obs_units**" options within the job command lines.

```
fcst_lev = [];
obs_lev  = [];
```

The user may specify a comma-separated list of forecast and observation level types to be used for any analyses to be performed. If multiple level types are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-fcst_lev**" and "**-obs_lev**" options within the job command lines.

```
obtype = [];
```

The user may specify a comma-separated list of observation types to be used for all analyses. If multiple observation types are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-obtype**" option within the job command line.

```
vx_mask = [];
```

The user may specify a comma-separated list of verification masking regions to be used for all analyses. If multiple verification masking regions are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-vx_mask**" option within the job command line.

```
interp_mthd = [];
```

The user may specify a comma-separated list of interpolation methods to be used for all analyses. If multiple interpolation methods are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-interp_mthd**" option within the job command line.

```
interp_pnts = [];
```

The user may specify a comma-separated list of interpolation points to be used for all analyses. If multiple interpolation points are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-interp_pnts**" option within the job command line.

```
fcst_thresh = [];
obs_thresh  = [];
cov_thresh  = [];
```

The user may specify comma-separated lists of forecast, observation, and coverage thresholds to be used for any analyses to be performed. If multiple thresholds are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-fcst_thresh**", "**-obs_thresh**", and "**-cov_thresh**" options within the job command lines.

```
alpha = [];
```

The user may specify a comma-separated list alpha confidence values to be used for all analyses. If alpha values are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-alpha**" option within the job command line.

```
line_type = [];
```

The user may specify a comma-separated list of line types to be used for all analyses. If multiple line types are listed, the analyses will be performed on their union. These selections may be further refined by using the "**-line_type**" option within the job command line.

```
column = [];
weight = [];
```

The column and weight fields are used to define a skill score index. The computation of a single value will be computed from each column and weight value specified. The GO Index is a specific example of a skill score index.

```
jobs = [
   "-job filter -dump_row ./filter_job.stat"
];
```

The user may specify one or more analysis jobs to be performed on the STAT lines that remain after applying the filtering parameters listed above. Each entry in the joblist contains the task and additional filtering options for a single analysis to be performed. The format for an analysis job is as follows:

-job job_name REQUIRED and OPTIONAL ARGUMENTS

All possible tasks for **job_name** are listed in Table 12.2.

Table 12.2: Description of components of the job command lines for the Stat-Analysis tool.

Job Name	**Job commandDescription**	**Required Arguments**
filter	Filters out the statistics lines based on applying options* (See note below table)	-dump_row
summary	Computes the mean, standard deviation, percentiles (min, 10th, 25th, 50th, 75th, 90th, and max), interquartile range, range, wmo_mean, and wmo_weighted_mean	-line_type -column
aggregate	Aggregates the statistics output, computing the statistic specified for the entire collection of valid lines	-line_type
aggregate_stat	Aggregates the statistics output, and converts the input line type to the output line type specified	-line_type -out_line_type
ss_index	Calculates a user-defined Skill Score index as described in section 12.2.5.	-model forecast -model reference
go_index	Calculates the GO Index as described in section 12.2.5.	-model forecast -model reference
ramp	Defines a ramp event on a time-series of forecast and observed values. The amount of change from one time to the next is computed for forecast and observed values. Those changes are thresholded to define events which are used to populate a 2x2 contingency table.	-ramp_type -ramp_thresh -out_line_type -column -ramp_time -ramp_exact -ramp_window

---

```
out_alpha = 0.05;
```

This entry specifies the alpha value to be used when computing confidence intervals for output statistics. It is similar to the **ci_alpha** entry describe in Section 3.5.1.

```
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" ];
```

These entries specify lists of statistics in the form LINE_TYPE:COLUMN to which the various WMO mean logic types should be applied for the summary job type.

```
vif_flag = FALSE;
```

The variance inflation factor (VIF) flag indicates whether to apply a first order variance inflation when calculating normal confidence intervals for an aggregated time series of contingency table counts or partial sums. The VIF adjusts the variance estimate for the lower effective sample size caused by autocorrelation of the statistics through time. A value of **FALSE** will not compute confidence intervals using the VIF. A value of **TRUE** will include the VIF, resulting in a slightly wider normal confidence interval.

The Stat-Analysis tool support several additional job command options which may be specified either on the command line when running a single job or within the **jobs** entry within the configuration file. These additional options are described below:

```
-by col_name
```

This job command option is extremely useful. It can used multiple times to specify a list of STAT header column names. When reading each input line, the Stat-Analysis tool concatenates together the entries in the specified columns and keeps track of the unique cases. It applies the logic defined for that job to each unique subset of data. For example, if your output was run over many different model names and masking regions, specify **-by MODEL,VX_MASK** to get output for each unique combination rather than having to run many very similar jobs.

```
-column_min    col_name value
-column_max    col_name value
-column_eq     col_name value
-column_thresh col_name thresh
-column_str    col_name string
```

The column filtering options may be used when the **-line_type** has been set to a single value. These options take two arguments, the name of the data column to be used followed by a value, string, or threshold to be applied. If multiple column_min/max/eq/thresh/str options are listed, the job will be performed on their intersection. Each input line is only retained if its value meets the numeric filtering criteria defined or matches one of the strings defined by the **-column_str** option. Multiple filtering strings may be listed using commas. Defining thresholds in MET is described in Section 3.5.1.

```
-dump_row file
```

Each analysis job is performed over a subset of the input data. Filtering the input data down to a desired subset is often an iterative process. The **-dump_row** option may be used for each job to specify the name of an output file to which the exact subset of data used for that job will be written. When initially constructing Stat-Analysis jobs, users are strongly encouraged to use the option and check it contents to ensure that the analysis was actually done over the intended subset.

```
-out_line_type name
```

This option specifies the desired output line type for the the **aggregate_stat** job type.

```
-out_stat file
-set_hdr  col_name string
```

The Stat-Analysis tool writes its output to either standard out or the file specified using the **-out** command line option. However that output lacks the standard STAT header columns. The **-out_stat** job command option may be used for each job to specify the name of an output file to which full STAT output lines should be written. Jobs will often combine output with multiple entries in the header columns. For example, a job may aggregate output with three different values in the **VX_MASK** column, such as "mask1", "mask2", and "mask3". The output **VX_MASK** column will contain the unique values encountered concatenated together with commas: "mask1,mask2,mask3". Alternatively, the **-set_hdr** option may be used to specify what should be written to the output header columns, such as "-set_hdr VX_MASK all_three_masks".

When using the "-out_stat" option to create a .stat output file and stratifying results using one or more "-by" job command options, those columns may be referenced in the "-set_hdr" option. When using mulitple "-by" options, use "CASE" to reference the full case information string:

```
-job aggregate_stat -line_type MPR -out_line_type CNT -by FCST_VAR,OBS_SID \
-set_hdr VX_MASK OBS_SID -set_hdr DESC CASE
```

The example above reads MPR lines, stratifies the data by forecast variable name and station ID, and writes the output for each case to a .stat output file. When creating the .stat file, write the full case information to the DESC output column and the station ID to the VX_MASK column.

```
-mask_grid name
-mask_poly file
-mask_sid  file|list
```

When processing input MPR lines, these options may be used to define a masking grid, polyline, or list of station ID's to filter the matched pair data geographically prior to computing statistics. The **-mask_sid** option is a station ID masking file or a comma-separated list of station ID's for filtering the matched pairs spatially. See the description of the "sid" entry in 3.5.1.

```
-out_fcst_thresh thresh
-out_obs_thresh  thresh
-out_thresh      thresh
-out_cnt_logic   string
```

When processing input MPR lines, these options are used to define the forecast, observation, or both thresholds to be applied when computing statistics. For categorical output line types (FHO, CTC, CTS, MCTC, MCTS) these define the categorical thresholds. For continuous output line types (SL1L2, SAL1L2, CNT), these define the continuous filtering thresholds and **-out_cnt_logic** defines how the forecast and observed logic should be combined.

```
-out_fcst_wind_thresh thresh
-out_obs_wind_thresh  thresh
-out_wind_thresh      thresh
-out_wind_logic       string
```

These job command options are analogous to the options listed above but apply when processing input MPR lines and deriving wind direction statistics.

```
-out_bin_size value
```

When processing input ORANK lines and writing output RHIST or PHIST lines, this option defines the output histogram bin width to be used.

12.3.3 stat-analysis tool output

The output generated by the Stat-Analysis tool contains statistics produced by the analysis. It also records information about the analysis job that produced the output for each line. Generally, the output is printed to the screen. However, it can be redirected to an output file using the "**-out**" option. The format of output from each STAT job command is described below.

The "-by column" job command option may be used to run the same job multiple times on unique subsets of data. Specify the "-by column" option one or more times to define a search

key, and that job will be run once for each unique search key found. For example, use "-by VX_MASK" to run the same job for multiple masking regions, and output will be generated for each unique masking region found. Use "-by VX_MASK -by FCST_LEAD" to generate output for each unique combination of masking region and lead time.

Job: filter

This job command finds and filters STAT lines down to those meeting criteria specified by the filter's options. The filtered STAT lines are written to a file specified by the "**-dump_row**" option.

The output of this job is the same STAT format described in sections 7.3.3, 8.3.3, and 10.3.3.

Job: summary

This job produces summary statistics for the column name and line type specified by the "**-column**" and "**-line_type**" options. The output of this job type consists of three lines. The first line contains "**JOB_LIST**", followed by a colon, then the filtering and job definition parameters used for this job. The second line contains "**COL_NAME**", followed by a colon, then the column names for the data in the next line. The third line contains the word "**SUMMARY**", followed by a colon, then the total, mean with confidence intervals, standard deviation with confidence intervals, minimum value, percentiles (10th, 25th, 50th, 75th, and 90th), the maximum value, the interquartile range, the range, and WMO mean information. The output columns are shown in Table 12.3 below.

Table 12.3: Columnar output of "summary" job output from the Stat-Analysis tool.

Column Number	Description
1	**SUMMARY**: (job type)
2	Total
3-7	Mean including normal and bootstrap upper and lower confidence limits
8-10	Standard deviation including bootstrap upper and lower confidence limits
11	Minimum value
12	10th percentile
13	25th percentile
14	Median (50th percentile)
15	75th percentile
16	90th percentile
17	Maximum value
18	Interquartile range (75th - 25th percentile)
19	Range (Maximum - Minimum)
20	WMO Mean type
21	WMO Unweighted Mean value
22	WMO Weighted Mean value

Job: aggregate

This job aggregates output from the STAT line type specified using the "**-line_type**" argument. The output of this job type is in the same format as the line type specified (see Sections 7.3.3, 8.3.3, and 10.3.3). Again the output consists of three lines. The first line contains "**JOB_LIST**", as described above. The second line contains "**COL_NAME**", followed by a colon, then the column names for the line type selected. The third line contains the name of the line type selected followed by the statistics for that line type.

Job: aggregate_stat

This job is similar to the "**aggregate**" job listed above, however the format of its output is determined by the "**-out_line_type**" argument. Again the output consists of three lines for "**JOB_LIST**", "**COL_NAME**", and the name of the output STAT line, as described above. Valid combinations of the "**-line_type**" and "**-out_line_type**" arguments are listed in Table 12.4 below.

Table 12.4: Valid combinations of "-line_type" and "-out_line_type" arguments for the "aggregate_stat" job.

Input Line Type	Output Line Type
FHO or CTC	CTS
MCTC	MCTS
SL1L2 or SAL1L2	CNT
VL1L2 or VAL1L2	WDIR (wind direction)
PCT	PSTD, PJC, PRC
NBRCTC	NBRCTS
ORANK	RHIST, PHIST, RELP, SSVAR
MPR	CNT, SL1L2, SAL1L2, WDIR
MPR	FHO, CTC, CTS, MCTC, MCTS, PCT, PSTD, PJC, or PRC (must specify "**-out_fcst_thresh**" and "**-out_obs_thresh**" arguments)

Job: ss_index

The output from this job consists of three lines, the first two of which contain "**JOB_LIST**" and "**COL_NAME**", as described above. The third line contains "**SS_INDEX**" followed by a colon and then the value computed for the user-defined Skill Score Index.

Job: go_index

The output from this job consists of three lines, the first two of which contain "**JOB_LIST**" and "**COL_NAME**", as described above. The third line contains "**GO_INDEX**" followed by a colon and then the value computed for the GO Index.

Job: ramp

The ramp job operates on a time-series of forecast and observed values and is analogous to the RIRW (Rapid Intensification and Weakening) job described in Section 21.3.3. The amount of change from one time to the next is computed for forecast and observed values. Those changes are thresholded to define events which are used to populate a 2x2 contingency table.

See the README file in the installed share/met/config directory for a detailed description of the job command options available for ramp job type.

The default output for this job is contingency table counts and statistics (**-out_line_type CTC,CTS**). Matched pair information may also be output by requesting MPR output (**-out_line_type CTC,CTS,MPR**).

Chapter 13

Series-Analysis Tool

13.1 Introduction

The Series-Analysis Tool accumulates statistics separately for each horizontal grid location over a series. Often, this series is over time or height, though any type of series is possible. This differs from the Grid-Stat tool in that Grid-Stat verifies all grid locations together as a group. Thus, the Series-Analysis Tool can be used to find verification information specific to certain locations or see how model performance varies over the domain.

13.2 Practical Information

This Series-Analysis tool performs verification of gridded model fields using matching gridded observation fields. It computes a variety of user-selected statistics. These statistics are a subset of those produced by the Grid-Stat tool, with options for statistic types, thresholds, and conditional verification options as discussed in the Chapter 8. However, these statistics are computed separately for each grid location and accumulated over some series such as time or height, rather than accumulated over the whole domain for a single time or height as is done by Grid-Stat.

This tool computes statistics for exactly one series each time it is run. Multiple series may be processed by running the tool multiple times. The length of the series to be processed is determined by the first of the following that is greater than one: the number of forecast fields in the configuration file, the number of observation fields in the configuration file, the number of input forecast files, the number of input observation files. Several examples of defining series are described below.

To define a time series of forecasts where the valid time changes for each time step, set the forecast and observation fields in the configuration file to single values and pass the tool multiple forecast and observation

files. The tool will loop over the forecast files, extract the specified field from each, and then search the observation files for a matching record with the same valid time.

To define a time series of forecasts that all have the same valid time, set the forecast and observation fields in the configuration file to single values. Pass the tool multiple forecast files and a single observation file containing the verifying observations. The tool will loop over the forecast files, extract the specified field from each, and then retrieve the verifying observations.

To define a series of vertical levels all contained in a single input file, set the forecast and observation fields to a list of the vertical levels to be used. Pass the tool single forecast and observation files containing the vertical level data. The tool will loop over the forecast field entries, extract that field from the input forecast file, and then search the observation file for a matching record.

13.2.1 series_analysis usage

The usage statement for the Series-Analysis tool is shown below:

```
Usage: series_analysis
       -fcst  file_1 ... file_n | fcst_file_list
       -obs   file_1 ... file_n | obs_file_list
       [-both file_1 ... file_n | both_file_list]
       [-paired]
       -out file
       -config file
       [-log file]
       [-v level]
       [-compress level]
```

series_analysis has four required arguments and accepts several optional ones.

Required arguments series_stat

1. The **-fcst file_1 ... file_n | fcst_file_list** options specify the gridded forecast files or ASCII files containing lists of file names to be used.

2. The **-obs file_1 ... file_n | obs_file_list** are the gridded observation files or ASCII files containing lists of file names to be used.

3. The **-out file** is the NetCDF output file containing computed statistics.

4. The **-config file** is a Series-Analysis Configuration file containing the desired settings.

Optional arguments for series_analysis

5. To set both the forecast and observations to the same set of files, use the optional **-both file_1 ... file_n | both_file_list** option to the same set of files. This is useful when reading the NetCDF matched pair output of the Grid-Stat tool which contains both forecast and observation data.

6. The **-paired** option indicates that the -fcst and -obs file lists are already paired, meaning there is a one-to-one correspondence between the files in those lists. This option affects how missing data is handled. When -paired is not used, missing or incomplete files result in a runtime error with no output file being created. When -paired is used, missing or incomplete files result in a warning with output being created using the available data.

7. The **-log file** outputs log messages to the specified file.

8. The **-v level** overrides the default level of logging (2).

9. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

An example of the series_analysis calling sequence is shown below:

```
series_analysis \
-fcst   myfcstfilelist.txt \
-obs    myobsfilelist.txt \
-config SeriesAnalysisConfig \
-out    out/my_series_statistics.nc
```

In this example, the Series-Analysis tool will process the list of forecast and observation files specified in the text file lists into statistics for each grid location using settings specified in the configuration file. Series-Analysis will create an output NetCDF file containing requested statistics.

13.2.2 series_analysis output

The Series-Analysis tool produces NetCDF files containing output statistics for each grid location from the input files. The details about the output statistics available from each output line type are detailed in Chapter 5 since they are also produced by the Grid-Stat Tool. A subset of these can be produced by this tool, with the most notable exceptions being the wind vector and neighborhood statistics. Users can inventory the contents of the Series-Analysis output files using the ncdump -h command to view header information. Additionally, ncview or the plot_data_plane tool can be used to visualize the output. An example of Series-Analysis output is shown in Figure 13.1 below.

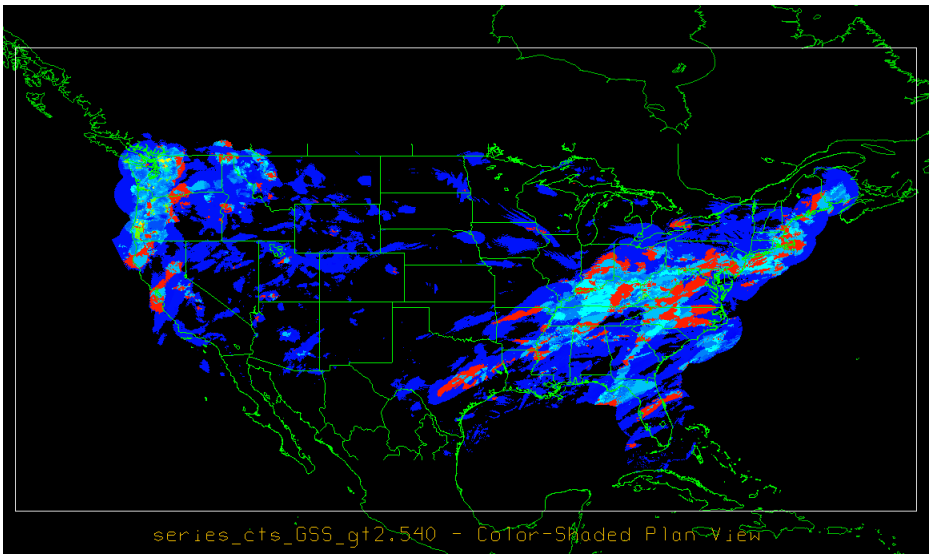

Figure 13.1: An example of the Gilbert Skill Score for precipitation forecasts at each grid location for a month of files.

13.2.3 series_analysis configuration file

The default configuration file for the Series-Analysis tool named *SeriesAnalysisConfig_default* can be found in the installed *share/met/config* directory. The contents of the configuration file are described in the subsections below.

Note that environment variables may be used when editing configuration files, as described in the Section 4.1.2 for the PB2NC tool.

```
model          = "WRF";
desc           = "NA";
obtype         = "ANALYS";
regrid         = { ... }
fcst           = { ... }
obs            = { ... }
climo_mean     = { ... }
climo_stdev    = { ... }
ci_alpha       = [ 0.05 ];
boot           = { interval = PCTILE; rep_prop = 1.0; n_rep = 1000;
                   rng = "mt19937"; seed = ""; }
mask           = { grid = [ "FULL" ]; poly = []; }
rank_corr_flag = TRUE;
tmp_dir        = "/tmp";
version        = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
block_size = 1024;
```

Number of grid points to be processed concurrently. Set smaller to use less memory but increase the number of passes through the data. The amount of memory the Series-Analysis tool consumes is determined by the size of the grid, the length of the series, and the **block_size** entry defined above. The larger this entry is set the faster the tool will run, subject to the amount of memory available on the machine.

```
vld_thresh = 1.0;
```

Ratio of valid matched pairs for the series of values at each grid point required to compute statistics. Set to a lower proportion to allow some missing values. Setting it to 1.0 requires that every data point be valid over the series to compute statistics.

```
output_stats = {
   fho    = [];
   ctc    = [];
   cts    = [];
   mctc   = [];
   mcts   = [];
   cnt    = ["RMSE", "FBAR", "OBAR"];
   sl1l2  = [];
   sal1l2 = [];
   pct    = [];
   pstd   = [];
   pjc    = [];
   prc    = [];
}
```

The **output_stats** array controls the type of output that the Series-Analysis tool generates. Each flag corresponds to an output line type in the STAT file and is used to specify the comma-separated list of statistics to be computed. Use the column names from the tables listed below to specify the statistics. The output flags correspond to the following types of output line types:

1. **FHO** for Forecast, Hit, Observation Rates (See Table 7.2)

2. **CTC** for Contingency Table Counts (See Table 7.3)
3. **CTS** for Contingency Table Statistics (See Table 7.4)
4. **MCTC** for Multi-Category Contingency Table Counts (See Table 7.8)
5. **MCTS** for Multi-Category Contingency Table Statistics (See Table 7.9)
6. **CNT** for Continuous Statistics (See Table 7.6)
7. **SL1L2** for Scalar L1L2 Partial Sums (See Table 7.15)
8. **SAL1L2** for Scalar Anomaly L1L2 Partial Sums climatological data is supplied (See Table 7.16)
9. **PCT** for Contingency Table Counts for Probabilistic forecasts (See Table 7.10)
10. **PSTD** for Contingency Table Statistics for Probabilistic forecasts (See Table 7.11)
11. **PJC** for Joint and Conditional factorization for Probabilistic forecasts (See Table 7.12)
12. **PRC** for Receiver Operating Characteristic for Probabilistic forecasts (See Table 7.13)

Chapter 14

Grid-Diag Tool

14.1 Introduction

The Grid-Diag tool creates histograms (probability distributions when normalized) for an arbitrary collection of data fields and levels. Joint histograms will be created for all possible pairs of variables. Masks can be used to subset the data fields spatially. The histograms are accumulated over a time series of input data files, similar to Series-Analysis.

14.2 Practical information

14.2.1 grid_diag usage

The following sections describe the usage statement, required arguments, and optional arguments for **grid_diag**.

```
Usage: grid_diag
       -data file_1 ... file_n | data_file_list
       -out file
       -config file
       [-log file]
       [-v level]
```

grid_diag has required arguments and can accept several optional arguments.

Required arguments for grid_diag

1. The **-data file_1 ... file_n | data_file_list** options specify the gridded data files or an ASCII file containing list of file names to be used.

2. The **-out** argument is the NetCDF output file.

3. The **-config file** is the configuration file to be used. The contents of the configuration file are discussed below.

Optional arguments for grid_diag

4. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

5. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

14.2.2 grid_diag configuration file

The default configuration file for the Grid-Diag tool named 'GridDiagConfig_default' can be found in the installed **share/met/config/ directory**. It is encouraged for users to copy these default files before modifying their contents. The contents of the configuration file are described in the subsections below.

```
model         = "GFS";
regrid        = { ... }
censor_thresh = [];
censor_val    = [];
mask          = { grid = [ "FULL" ]; poly = []; }
version       = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
data = {
   field = [
      {
         name   = "APCP";
         level  = ["L0"];
         n_bins = 30;
         range  = [0, 12];
```

```
        },
        {
           name   = "PWAT";
           level  = ["L0"];
           n_bins = 35;
           range  = [35, 70];
        }
     ];
  }
```

The **name** and **level** entries in the **data** dictionary define the data to be processed. The **n_bins** parameter specifies the number of histogram bins for that variable, and the **range** parameter the lower and upper bounds of the histogram. The interval length is the upper and lower difference divided by **n_bins**.

14.2.3 grid_diag output file

The NetCDF file has a dimension for each of the specified data variable and level combinations, e.g. APCP_L0 and PWAT_L0. The bin minimum, midpoint, and maximum values are indicated with an _min, _min, or _max appended to the variable/level.

For each variable/level combination in the data dictionary a corresponding histogram will be output to the NetCDF file. For example, hist_APCP_L0 and hist_PWAT_L0. These are the counts of all data values falling within the bin. Data values below the minimum or above the maximum are included in the lowest and highest bins, respectively. In addition to 1D histograms, 2D histograms for all variable/level pairs are written. For example, hist_APCP_L0_PWAT_L0 is the joint histogram for those two variables/levels.

Chapter 15

MODE Tool

15.1 Introduction

This chapter provides a description of the Method for Object-Based Diagnostic Evaluation (MODE) tool, which was developed at the Research Applications Laboratory, NCAR/Boulder, USA. More information about MODE can be found in Davis *et al.* (2006a, b), Brown *et al.* (2007) and Bullock *et al.* (2016).

MODE was developed in response to a need for verification methods that can provide diagnostic information that is more directly useful and meaningful than the information that can be obtained from traditional verification approaches, especially in application to high-resolution NWP output. The MODE approach was originally developed for application to spatial precipitation forecasts, but it can also be applied to other fields with coherent spatial structures (*e.g.*, clouds, convection).

MODE is only one of a number of different approaches that have been developed in recent years to meet these needs. In the future, we expect that the MET package will include additional methods. References for many of these methods are provided at `http://www.rap.ucar.edu/projects/icp/index.html`.

MODE may be used in a generalized way to compare any two fields. For simplicity, field1 may be thought of in this chapter as "the forecast," while field2 may be thought of as "the observation", which is usually a gridded analysis of some sort. The convention of field1/field2 is also used in Table 15.2. MODE resolves objects in both the forecast and observed fields. These objects mimic what humans would call "regions of interest." Object attributes are calculated and compared, and are used to associate ("merge") objects within a single field, as well as to "match" objects between the forecast and observed fields. Finally, summary statistics describing the objects and object pairs are produced. These statistics can be used to identify correlations and differences among the objects, leading to insights concerning forecast strengths and weaknesses.

15.2 Scientific and statistical aspects

The methods used by the MODE tool to identify and match forecast and observed objects are briefly described in this section.

15.2.1 Resolving objects

The process used for resolving objects in a raw data field is called *convolution thresholding*. The raw data field is first convolved with a simple filter function as follows:

$$C(x,y) = \sum_{u,v} \phi(u,v)\, f(x-u, y-v).$$

In this formula, f is the raw data field, ϕ is the filter function, and C is the resulting convolved field. The variables (x,y) and (u,v) are grid coordinates. The filter function ϕ is a simple circular filter determined by a radius of influence R, and a height H:

$$\phi(x,y) = \begin{cases} H & \text{if } x^2 + y^2 \leq R^2 \\ 0 & \text{otherwise.} \end{cases}$$

The parameters R and H are not independent. They are related by the requirement that the integral of ϕ over the grid be unity:

$$\pi R^2 H = 1.$$

Thus, the radius of influence R is the only tunable parameter in the convolution process. Once R is chosen, H is determined by the above equation.

Once the convolved field C is in hand, it is thresholded to create a mask field M:

$$M(x,y) = \begin{cases} 1 & \text{if } C(x,y) \geq T \\ 0 & \text{otherwise.} \end{cases}$$

where T is the threshold. The objects are the connected regions where $M = 1$. Finally, the raw data are restored to object interiors to obtain the object field F:

$$F(x,y) = M(x,y) f(x,y).$$

Thus, two parameters — the radius of influence R, and the threshold T — control the entire process of resolving objects in the raw data field.

An example of the steps involved in resolving objects is shown in Figure 15.1. Figure 15.1a shows a "raw" precipitation field, where the vertical coordinate represents the precipitation amount. Part b shows the convolved field, and part c shows the masked field obtained after the threshold is applied. Finally, Figure 15.1d shows the objects once the original precipitation values have been restored to the interiors of the objects.

15.2.2 Attributes

Object attributes are defined both for single objects and for object pairs. One of the objects in a pair is from the forecast field and the other is taken from the observed field.

Area is simply a count of the number of grid squares an object occupies. If desired, a true area (say, in km^2) can be obtained by adding up the true areas of all the grid squares inside an object, but in practice this is seldom necessary.

Moments are used in the calculation of several object attributes. If we define $\xi(x, y)$ to be 1 for points (x, y) inside our object, and zero for points outside, then the first-order moments, S_x and S_y, are defined as

$$S_x = \sum_{x,y} x\,\xi(x,y) \qquad \text{and} \qquad S_y = \sum_{x,y} y\,\xi(x,y).$$

Higher order moments are similarly defined and are used in the calculation of some of the other attributes. For example, the **centroid** is a kind of geometric center of an object, and can be calculated from first moments. It allows one to assign a single point location to what may be a large, extended object.

Axis Angle, denoted by θ, is calculated from the second-order moments. It gives information on the orientation or "tilt" of an object. **Curvature** is another attribute that uses moments in its calculation, specifically, third-order moments.

Aspect Ratio is computed by fitting a rectangle around an object. The rectangle is aligned so that it has the same axis angle as the object, and the length and width are chosen so as to just enclose the object. We make no claim that the rectangle so obtained is the smallest possible rectangle enclosing the given object. However, this rectangle is much easier to calculate than a smallest enclosing rectangle and serves our purposes just as well. Once the rectangle is determined, the aspect ratio of the object is defined to be the width of the fitted rectangle divided by its length.

Another object attribute defined by MODE is **complexity**. Complexity is defined by comparing the area of an object to the area of its convex hull.

All the attributes discussed so far are defined for single objects. Once these are determined, they can be used to calculate attributes for pairs of objects. One example is **centroid difference**. This measure is simply the (vector) difference between the centroids of the two objects. Another example is **angle difference**, the difference between the axis angles.

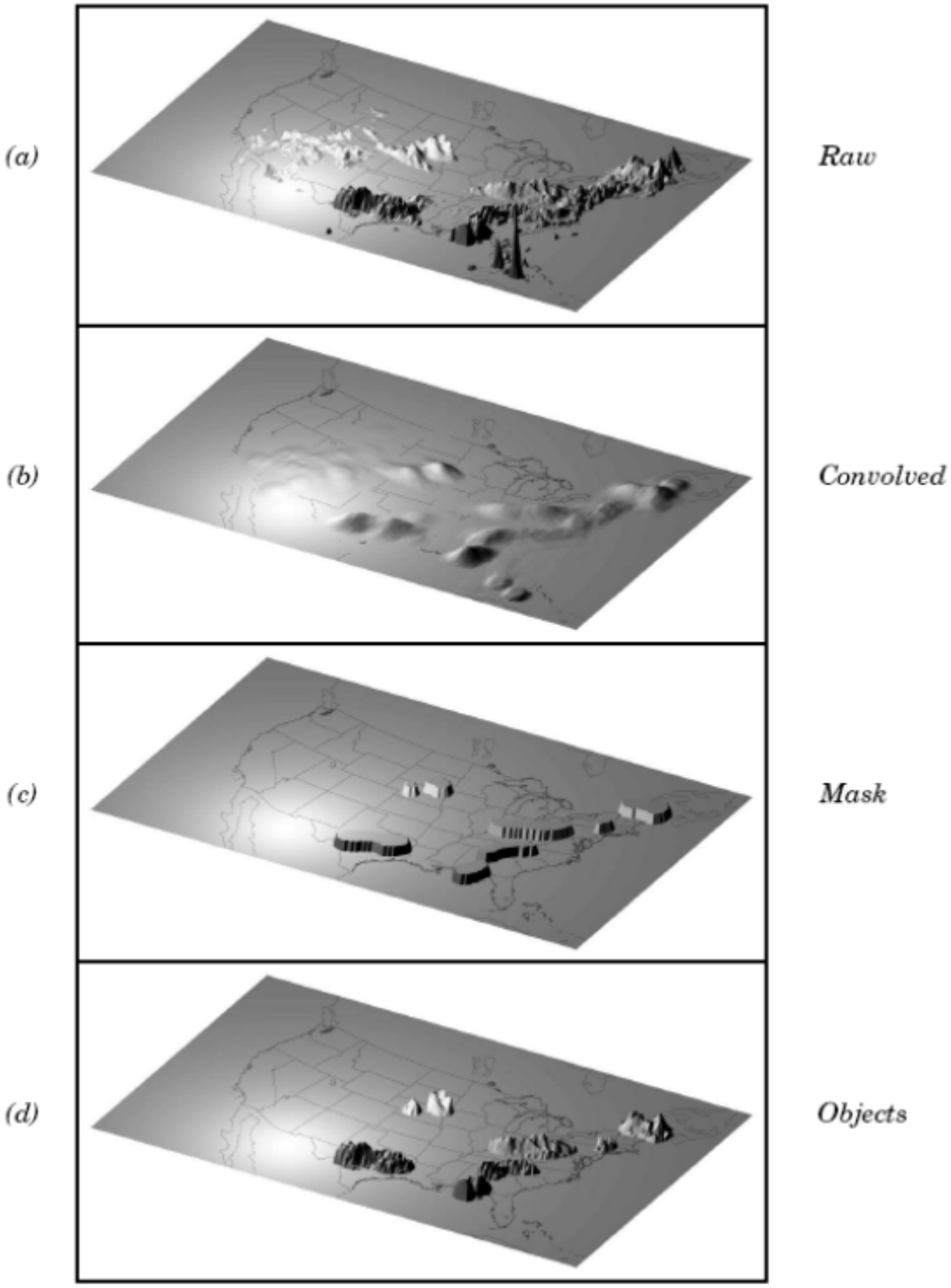

Figure 15.1: Example of an application of the MODE object identification process to a model precipitation field.

Several area measures are also used for pair attributes. **Union Area** is the total area that is in either one (or both) of the two objects. **Intersection Area** is the area that is inside both objects simultaneously. **Symmetric Difference** is the area inside at least one object, but not inside both.

15.2.3 Fuzzy logic

Once object attributes $\alpha_1, \alpha_2, \ldots, \alpha_n$ are estimated, some of them are used as input to a fuzzy logic engine that performs the matching and merging steps. **Merging** refers to grouping together objects in a single field, while **matching** refers to grouping together objects in different fields, typically the forecast and observed fields. Interest maps I_i are applied to the individual attributes α_i to convert them into interest values, which range from zero (representing no interest) to one (high interest). For example, the default interest map for centroid difference is one for small distances, and falls to zero as the distance increases. For other attributes (*e.g.*, intersection area), low values indicate low interest, and high values indicate more interest.

The next step is to define confidence maps C_i for each attribute. These maps (again with values ranging from zero to one) reflect how confident we are in the calculated value of an attribute. The confidence maps generally are functions of the entire attribute vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, in contrast to the interest maps, where each I_i is a function only of α_i. To see why this is necessary, imagine an electronic anemometer that outputs a stream of numerical values of wind speed and direction. It is typically the case for such devices that when the wind speed becomes small enough, the wind direction is poorly resolved. The wind must be at least strong enough to overcome friction and turn the anemometer. Thus, in this case, our confidence in one attribute (wind direction) is dependent on the value of another attribute (wind speed). In MODE, all of the confidence maps except the map for axis angle are set to a constant value of 1. The axis angle confidence map is a function of aspect ratio, with values near one having low confidence, and values far from one having high confidence.

Next, scalar weights $\mathbf{w}_i$ are assigned to each attribute, representing an empirical judgment regarding the relative importance of the various attributes. As an example, the initial development of MODE, centroid distance was weighted more heavily than other attributes, because the location of storm systems close to each other in space seemed to be a strong indication (stronger than that given by any other attribute) that they were related.

Finally, all these ingredients are collected into a single number called the total interest, $\mathbf{T}$, given by:

$$T(\alpha) = \frac{\sum_i w_i C_i(\alpha) I_i(\alpha_i)}{\sum_i w_i C_i(\alpha).}$$

This total interest value is then thresholded, and pairs of objects that have total interest values above the threshold are merged (if they are in the same field) or matched (if they are in different fields).

Another merging method is available in MODE, which can be used instead of, or along with, the fuzzy logic based merging just described. Recall that the convolved field is thresholded to produce the mask field. A second (lower) threshold can be specified so that objects that are separated at the higher threshold but joined at the lower threshold are merged.

15.2.4 Summary statistics

Once MODE has been run, summary statistics are written to an output file. These files contain information about all single and cluster objects and their attributes. Total interest for object pairs is also output, as are percentiles of intensity inside the objects. The output file is in a simple flat ASCII tabular format (with one header line) and thus should be easily readable by just about any programming language, scripting language, or statistics package. Refer to Section 15.3.3 for lists of the statistics included in the MODE output files. Example scripts will be posted on the MET website in the future.

15.3 Practical information

This section contains a description of how MODE can be configured and run. The MODE tool is used to perform a features-based verification of gridded model data using gridded observations. The input gridded model and observation datasets must be in one of the MET supported gridded file formats. The requirement of having all gridded fields using the same grid specification has been removed with METv5.1. The Grid-Stat tool performs no interpolation when the input model, observation, and climatology datasets must be on a common grid. MET will interpolate these files to a common grid if one is specified. There is a regrid option in the configuration file that allows the user to define the grid upon which the scores will be computed. The gridded analysis data may be based on observations, such as Stage II or Stage IV data for verifying accumulated precipitation, or a model analysis field may be used. However, users are cautioned that it is generally unwise to verify model output using an analysis field produced by the same model.

MODE provides the capability to select a single model variable/level from which to derive objects to be analyzed. MODE was developed and tested using accumulated precipitation. However, the code has been generalized to allow the use of any gridded model and observation field. Based on the options specified in the configuration file, MODE will define a set of simple objects in the model and observation fields. It will then compute an interest value for each pair of objects across the fields using a fuzzy engine approach. Those interest values are thresholded, and any pairs of objects above the threshold will be matched/merged. Through the configuration file, MODE offers a wide range of flexibility in how the objects are defined, processed, matched, and merged.

15.3.1 mode usage

The usage statement for the MODE tool is listed below:

```
Usage: mode
       fcst_file
       obs_file
       config_file
       [-config_merge merge_config_file]
```

```
[-outdir path]
[-log file]
[-v level]
[-compress level]
```

The MODE tool has three required arguments and can accept several optional arguments.

Required arguments for mode

1. The **fcst_file** argument indicates the gridded file containing the model field to be verified.

2. The **obs_file** argument indicates the gridded file containing the gridded observations to be used for the verification of the model.

3. The **config_file** argument indicates the name of the configuration file to be used. The contents of the configuration file are discussed below.

Optional arguments for mode

4. The **-config_merge merge_config_file** option indicates the name of a second configuration file to be used when performing fuzzy engine merging by comparing the model or observation field to itself. The MODE tool provides the capability of performing merging within a single field by comparing the field to itself. Interest values are computed for each object and all of its neighbors. If an object and its neighbor have an interest value above some threshold, they are merged. The **merge_config_file** controls the settings of the fuzzy engine used to perform this merging step. If a **merge_config_file** is not provided, the configuration specified by the config_file in the previous argument will be used.

5. The **-outdir path** option indicates the directory where output files should be written.

6. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

7. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

8. The **-compress level** option indicates the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. The value of "level" will override the default setting of 0 from the configuration file or the environment variable MET_NC_COMPRESS. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

An example of the MODE calling sequence is listed below:

Example 1:

```
mode sample_fcst.grb \
sample_obs.grb \
MODEConfig_grb
```

In Example 1, the MODE tool will verify the model data in the sample_fcst.grb GRIB file using the observations in the sample_obs.grb GRIB file applying the configuration options specified in the MODEConfig_grb file.

A second example of the MODE calling sequence is presented below:

Example 2:

```
mode sample_fcst.nc \
sample_obs.nc \
MODEConfig_nc
```

In Example 2, the MODE tool will verify the model data in the sample_fcst.nc NetCDF output of pcp_combine using the observations in the sample_obs.nc NetCDF output of pcp_combine, using the configuration options specified in the MODEConfig_nc file. Since the model and observation files contain only a single field of accumulated precipitation, the MODEConfig_nc file should specify that accumulated precipitation be verified.

15.3.2 mode configuration file

The default configuration file for the MODE tool, MODEConfig_default, can be found in the installed share/met/config directory. Another version of the configuration file is provided in scripts/config. We encourage users to make a copy of the configuration files prior to modifying their contents. Descriptions of MODEConfig_default and the required variables for any MODE configuration file are also provided below. While the configuration file contains many entries, most users will only need to change a few for their use. Specific options are described in the following subsections.

Note that environment variables may be used when editing configuration files, as described in Section 4.1.2 for the PB2NC tool.

```
model          = "WRF";
desc           = "NA";
obtype         = "ANALYS";
regrid         = { ... }
met_data_dir   = "MET_BASE";
output_prefix  = "";
version        = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
grid_res = 4;
```

The **grid_res** entry is the nominal spacing for each grid square in kilometers. This entry is not used directly in the code, but subsequent entries in the configuration file are defined in terms of it. Therefore, setting this appropriately will help ensure that appropriate default values are used for these entries.

```
quilt = FALSE;
```

The **quilt** entry indicates whether all permutations of convolution radii and thresholds should be run.

- If **FALSE**, the number of forecast and observation convolution radii and thresholds must all match. One configuration of MODE will be run for each group of settings in those lists.
- If **TRUE**, the number of forecast and observation convolution radii must match and the number of forecast and observation convolution thresholds must match. For N radii and M thresholds, NxM configurations of MODE will be run.

```
fcst = {
   field = {
      name = "APCP";
      level = "A03";
   }
   censor_thresh      = [];
   censor_val         = [];
   conv_radius        = 60.0/grid_res; // in grid squares
   conv_thresh        = >=5.0;
   vld_thresh         = 0.5;
   filter_attr_name   = [];
   filter_attr_thresh = [];
   merge_thresh       = >=1.25;
   merge_flag         = THRESH;
}
obs = fcst;
```

The **field** entries in the forecast and observation dictionaries specify the model and observation variables and level to be compared. See a more complete description of them in Section 3.5.1. In the above example, the forecast settings are copied into the observation dictionary using **obs = fcst;.**

The **censor_thresh** and **censor_val** entries are used to censor the raw data as described in Section 3.5.1. Their functionality replaces the **raw_thresh** entry, which is deprecated in met-6.1. Prior to defining objects, it is recommended that the raw fields should be made to look similar to each other. For example, if the model only predicts values for a variable above some threshold, the observations should be thresholded at that same level. The censor thresholds can be specified using symbols. By default, no censor thresholding is applied.

The **conv_radius** entry defines the radius of the circular convolution applied to smooth the raw fields. The radii are specified in terms of grid units. The default convolution radii are defined in terms of the previously defined **grid_res** entry. Multiple convolution radii may be specified as an array (e.g. **conv_radius = [5, 10, 15];**).

The **conv_thresh** entry specifies the threshold values to be applied to the convolved field to define objects. By default, objects are defined using a convolution threshold of 5.0. Multiple convolution thresholds may be specified as an array (e.g. **conv_thresh = [>=5.0, >=10.0, >=15.0];**).

Multiple convolution radii and thresholds and processed using the logic defined by the **quilt** entry.

The **vld_thresh** entry must be set between 0 and 1. When performing the circular convolution step if the proportion of bad data values in the convolution area is greater than or equal to this threshold, the resulting convolved value will be bad data. If the proportion is less than this threshold, the convolution will be performed on only the valid data. By default, the **vld_thresh** is set to 0.5.

The **filter_attr_name** and **filter_attr_thresh** entries are arrays of the same length which specify object filtering criteria. By default, no object filtering criteria is defined.

The **filter_attr_name** entry is an array of strings specifying the MODE output header column names for the object attributes of interest, such as **AREA**, **LENGTH**, **WIDTH**, and **INTENSITY_50**. In addition, **ASPECT_RATIO** specifies the aspect ratio (width/length), **INTENSITY_101** specifies the mean intensity value, and **INTENSITY_102** specifies the sum of the intensity values.

The **filter_attr_thresh** entry is an array of thresholds for these object attributes. Any simple objects not meeting all of the filtering criteria are discarded.

Note that the **area_thresh** and **inten_perc_thresh** entries from earlier versions of MODE are replaced by these options and are now deprecated.

The **merge_thresh** entry is used to define larger objects for use in merging the original objects. It defines the threshold value used in the double thresholding merging technique. Note that in order to use this merging technique, it must be requested for both the forecast and observation fields. These thresholds should be chosen to define larger objects that fully contain the originally defined objects. For example, for objects defined as >=5.0, a merge threshold of >=2.5 will define larger objects that fully contain the original

objects. Any two original objects contained within the same larger object will be merged. By default, the merge thresholds are set to be greater than or equal to 1.25. Multiple merge thresholds may be specified as an array (e.g. **merge_thresh = [>=1.0, >=2.0, >=3.0];**). The number of **merge_thresh** entries must match the number of **conv_thresh** entries.

The **merge_flag** entry controls what type of merging techniques will be applied to the objects defined in each field.

- **NONE** indicates that no merging should be applied.
- **THRESH** indicates that the double thresholding merging technique should be applied.
- **ENGINE** indicates that objects in each field should be merged by comparing the objects to themselves using a fuzzy engine approach.
- **BOTH** indicates that both techniques should be used.

By default, the double thresholding merging technique is applied.

```
mask_missing_flag = NONE;
```

The **mask_missing_flag** entry specifies how missing data in the raw model and observation fields will be treated.

- **NONE** indicates no additional processing is to be done.
- **FCST** indicates missing data in the observation field should be used to mask the forecast field.
- **OBS** indicates missing data in the forecast field should be used to mask the observation field.
- **BOTH** indicates masking should be performed in both directions (*i.e.*, mask the forecast field with the observation field and vice-versa).

Prior to defining objects, it is recommended that the raw fields be made to look similar to each other by assigning a value of BOTH to this parameter. However, by default no masking is performed.

```
match_flag = MERGE_BOTH;
```

The **match_flag** entry controls how matching will be performed when comparing objects from the forecast field to objects from the observation field. An interest value is computed for each possible pair of forecast/observation objects. The interest values are then thresholded to define which objects match. If two objects in one field happen to match the same object in the other field, then those two objects could be merged. The **match_flag** entry controls what type of merging is allowed in this context.

- **NONE** indicates that no matching should be performed between the fields at all.
- **MERGE_BOTH** indicates that additional merging is allowed in both fields.
- **MERGE_FCST** indicates that additional merging is allowed only in the forecast field.
- **NO_MERGE** indicates that no additional merging is allowed in either field, meaning that each object will match at most one object in the other field.

By default, additional merging is allowed in both fields.

```
max_centroid_dist = 800/grid_res;
```

Computing the attributes for all possible pairs of objects can take some time depending on the numbers of objects. The **max_centroid_dist** entry is used to specify how far apart objects should be in order to conclude that they have no chance of matching. No pairwise attributes are computed for pairs of objects whose centroids are farther away than this distance, defined in terms of grid units. Setting this entry to a reasonable value will improve the execution time of the MODE tool. By default, the maximum centroid distance is defined in terms of the previously defined **grid_res** entry.

```
mask = {
   grid = "";
   grid_flag = NONE; // Apply to NONE, FCST, OBS, or BOTH
   poly = "";
   poly_flag = NONE; // Apply to NONE, FCST, OBS, or BOTH
}
```

Defining a **grid** and **poly** masking region is described in Section 3.5.1. Applying a masking region when running MODE sets all grid points falling outside of that region to missing data, effectively limiting the area of which objects should be defined.

The **grid_flag** and **poly_flag** entries specify how the grid and polyline masking should be applied:

- **NONE** indicates that the masking grid should not be applied.
- **FCST** indicates that the masking grid should be applied to the forecast field.
- **OBS** indicates that the masking grid should be applied to the observation field.
- **BOTH** indicates that the masking grid should be applied to both fields.

By default, no masking grid or polyline is applied.

```
weight = {
   centroid_dist    = 2.0;
   boundary_dist    = 4.0;
   convex_hull_dist = 0.0;
   angle_diff       = 1.0;
   aspect_diff      = 0.0;
   area_ratio       = 1.0;
   int_area_ratio   = 2.0;
   curvature_ratio  = 0.0;
   complexity_ratio = 0.0;
   inten_perc_ratio = 0.0;
   inten_perc_value = 50;
}
```

The **weight** entries listed above control how much weight is assigned to each pairwise attribute when computing a total interest value for object pairs. The weights listed above correspond to the **centroid distance** between the objects, the **boundary distance** (or minimum distance), the **convex hull distance** (or minimum distance between the convex hulls of the objects), the **orientation angle difference**, the **aspect ratio difference**, the **object area ratio** (minimum area divided by maximum area), the **intersection divided by the minimum object area ratio**, the **curvature ratio**, the **complexity ratio**, and the **intensity ratio**. The weights need not sum to any particular value. When the total interest value is computed, the weighted sum is normalized by the sum of the weights listed above.

The **inten_perc_value** entry corresponds to the **inten_perc_ratio**. The **inten_perc_value** should be set between 0 and 102 to define which percentile of intensity should be compared for pairs of objects. 101 and 102 specify the intensity mean and sum, respectively. By default, the 50th percentile, or median value, is chosen.

```
interest_function = {
   centroid_dist      = ( ... );
   boundary_dist      = ( ... );
   convex_hull_dist   = ( ... );
   angle_diff         = ( ... );
   aspect_diff        = ( ... );
   corner             = 0.8;
   ratio_if           = ( ( 0.0, 0.0 ) ( corner, 1.0 ) ( 1.0, 1.0 ) );
   area_ratio         = ratio_if;
```

```
   int_area_ratio     = ( ... );
   curvature_ratio    = ratio_if;
   complexity_ratio   = ratio_if;
   inten_perc_ratio   = ratio_if;
}
```

The set of interest function entries listed above define which values are of interest for each pairwise attribute measured. The interest functions may be defined as a piecewise linear function or as an algebraic expression. A piecewise linear function is defined by specifying the corner points of its graph. An algebraic function may be defined in terms of several built-in mathematical functions. See Section 16.2for how interest values are used by the fuzzy logic engine. By default, many of these functions are defined in terms of the previously defined **grid_res** entry.

```
total_interest_thresh = 0.7;
```

The **total_interest_thresh** entry should be set between **0** and **1**. This threshold is applied to the total interest values computed for each pair of objects. Object pairs that have an interest value that is above this threshold will be matched, while those with an interest value that is below this threshold will remain unmatched. Increasing the threshold will decrease the number of matches while decreasing the threshold will increase the number of matches. By default, the total interest threshold is set to 0.7.

```
print_interest_thresh = 0.0;
```

The **print_interest_thresh** entry determines which pairs of object attributes will be written to the output object attribute ASCII file. The user may choose to set the **print_interest_thresh** to the same value as the **total_interest_thresh**, meaning that only object pairs that actually match are written to the output file. By default, the print interest threshold is set to zero, meaning that all object pair attributes will be written as long as the distance between the object centroids is less than the **max_centroid_dist** entry.

```
fcst_raw_plot = {
   color_table = "MET_BASE/colortables/met_default.ctable";
   plot_min = 0.0;
   plot_max = 0.0;
   colorbar_spacing = 1;
}
obs_raw_plot = {
```

```
   color_table = "MET_BASE/colortables/met_default.ctable";
   plot_min = 0.0;
   plot_max = 0.0;
   colorbar_spacing = 1;
}
object_plot = {
   color_table = "MET_BASE/colortables/mode_obj.ctable";
}
```

Specifying dictionaries to define the **color_table**, **plot_min**, and **plot_max** entries are described in Section 3.5.1.

The MODE tool generates a color bar to represent the contents of the colortable that was used to plot a field of data. The number of entries in the color bar matches the number of entries in the color table. The values defined for each color in the color table are also plotted next to the color bar. The **colorbar_spacing** entry is used to define the frequency with which the color table values should be plotted. Setting this entry to 1, as shown above, indicates that every color table value should be plotted. Setting it to an integer, n $>$ 1, indicates that only every n-th color table value should be plotted.

```
plot_valid_flag = FALSE;
```

When applied, the **plot_valid_flag** entry indicates that only the region containing valid data after masking is applied should be plotted.

- **FALSE** indicates the entire domain should be plotted.
- **TRUE** indicates only the region containing valid data after masking should be plotted.

The default value of this flag is FALSE.

```
plot_gcarc_flag = FALSE;
```

When applied, the **plot_gcarc_flag** entry indicates that the edges of polylines should be plotted using great circle arcs as opposed to straight lines in the grid. The default value of this flag is FALSE.

```
ps_plot_flag  = TRUE;
ct_stats_flag = TRUE;
```

These flags can be set to TRUE or FALSE to produce additional output, in the form of PostScript plots and contingency table counts and statistics, respectively.

```
nc_pairs_flag = {
   latlon     = TRUE;
   raw        = TRUE;
   object_raw = TRUE;
   object_id  = TRUE;
   cluster_id = TRUE;
   polylines  = TRUE;
}
```

Each component of the pairs information in the NetCDF file can be turned on or off. The old syntax is still supported: **TRUE** means accept the defaults, **FALSE** means no NetCDF output is generated. NetCDF output can also be turned off by setting all the individual dictionary flags to false.

```
shift_right = 0;
```

When MODE is run on global grids, this parameter specifies how many grid squares to shift the grid to the right. MODE does not currently connect objects from one side of a global grid to the other, potentially causing objects straddling the "cut" longitude to be separated into two objects. Shifting the grid by integer number of grid units enables the user to control where that longitude cut line occurs.

15.3.3 mode output

MODE produces output in ASCII, NetCDF, and PostScript formats.

ASCII output

The MODE tool creates two ASCII output files. The first ASCII file contains contingency table counts and statistics for comparing the forecast and observation fields. This file consists of 4 lines. The first is a header line containing column names. The second line contains data comparing the two raw fields after any masking of bad data or based on a grid or lat/lon polygon has been applied. The third contains data comparing the two fields after any raw thresholds have been applied. The fourth, and last, line contains data comparing the derived object fields scored using traditional measures.

Table 15.1: Format of MODE CTS output file.

MODE ASCII CONTINGENCY TABLE OUTPUT FORMAT		
Column Number	**MODE CTS Column Name**	**Description**
1	VERSION	Version number
2	MODEL	User provided text string designating model name
3	N_VALID	Number of valid data points
4	GRID_RES	User provided nominal grid resolution
5	DESC	User provided text string describing the verification task
6	FCST_LEAD	Forecast lead time in HHMMSS format
7	FCST_VALID	Forecast valid start time in YYYYMMDD_HHMMSS format
8	FCST_ACCUM	Forecast accumulation time in HHMMSS format
9	OBS_LEAD	Observation lead time in HHMMSS format; when field2 is actually an observation, this should be "000000"
10	OBS_VALID	Observation valid start time in YYYYMMDD_HHMMSS format
11	OBS_ACCUM	Observation accumulation time in HHMMSS format
12	FCST_RAD	Forecast convolution radius in grid squares
13	FCST_THR	Forecast convolution threshold
14	OBS_RAD	Observation convolution radius in grid squares
15	OBS_THR	Observation convolution threshold
16	FCST_VAR	Forecast variable
17	FCST_UNITS	Units for model variable
18	FCST_LEV	Forecast vertical level
19	OBS_VAR	Observation variable
20	OBS_UNITS	Units for observation variable
21	OBS_LEV	Observation vertical level
22	OBTYPE	User provided observation type
23	FIELD	Field type for this line:* RAW for the raw input fields * OBJECT for the resolved object fields
24	TOTAL	Total number of matched pairs
25	FY_OY	Number of forecast yes and observation yes
26	FY_ON	Number of forecast yes and observation no
27	FN_OY	Number of forecast no and observation yes
28	FN_ON	Number of forecast no and observation no
29	BASER	Base rate
30	FMEAN	Forecast mean
31	ACC	Accuracy
32	FBIAS	Frequency Bias
33	PODY	Probability of detecting yes
34	PODN	Probability of detecting no
35	POFD	Probability of false detection
36	FAR	False alarm ratio
37	CSI	Critical Success Index
38	GSS	Gilbert Skill Score
39	HK	Hanssen-Kuipers Discriminant
40	HSS	Heidke Skill Score
41	ODDS	Odds Ratio

This first file uses the following naming convention:

```
mode_PREFIX_FCST_VAR_LVL_vs_OBS_VAR_LVL_HHMMSSL_YYYYMMDD_HHMMSSV_HHMMSSA_cts.txt
```

where `PREFIX` indicates the user-defined output prefix, `FCST_VAR_LVL` is the forecast variable and vertical level being used, `OBS_VAR_LVL` is the observation variable and vertical level being used, `HHMMSSL` indicates the forecast lead time, `YYYYMMDD_HHMMSSV` indicates the forecast valid time, and `HHMMSSA` indicates the accumulation period. The `cts` string stands for contingency table statistics. The generation of this file can be disabled using the `ct_stats_flag` option in the configuration file. This CTS output file differs somewhat from the CTS output of the Point-Stat and Grid-Stat tools. The columns of this output file are summarized in Table 15.1.

The second ASCII file the MODE tool generates contains all of the attributes for simple objects, the merged cluster objects, and pairs of objects. Each line in this file contains the same number of columns, though those columns not applicable to a given line contain fill data. The first row of every MODE object attribute file is a header containing the column names. The number of lines in this file depends on the number of objects defined. This file contains lines of 6 types that are indicated by the contents of the **OBJECT_ID** column. The **OBJECT_ID** can take the following 6 forms: **FNN**, **ONN**, **FNNN_ONNN**, **CFNNN**, **CONNN**, **CFNNN_CONNN**. In each case, **NNN** is a three-digit number indicating the object index. While all lines have the first 18 header columns in common, these 6 forms for **OBJECT_ID** can be divided into two types - one for single objects and one for pairs of objects. The single object lines (**FNN**, **ONN**, **CFNNN**, and **CONNN**) contain valid data in columns 19–39 and fill data in columns 40–51. The object pair lines (**FNNN_ONNN** and **CFNNN_CONNN)** contain valid data in columns 40–51 and fill data in columns 19–39. These object identifiers are described in Table 15.2.

Table 15.2: Object identifier descriptions for MODE object attribute output files.

mode ASCII OBJECT IDENTIFIER DESCRIPTIONS		
Object identifier (object_id)	**Valid Data Columns**	**Description of valid data**
FNNN, ONNN	1-18,19-39	Attributes for simple forecast, observation objects
FNNN_ ONNN	1-18, 40-51	Attributes for pairs of simple forecast and observation objects
CFNNN, CONNN	1-18,19-39	Attributes for merged cluster objects in forecast, observation fields
CFNNN_ CONNN	1-18, 40-51	Attributes for pairs of forecast and observation cluster objects

A note on terminology: a cluster (referred to as "composite" in earlier versions) object need not necessarily consist of more than one simple object. A cluster object is by definition any set of one or more objects in one field which match a set of one or more objects in the other field. When a single simple forecast object matches a single simple observation object, they are each considered to be cluster objects as well.

The contents of the columns in this ASCII file are summarized in Table 15.3 and 15.4.

Table 15.3: Format of MODE object attribute output files.

mode ASCII OBJECT ATTRIBUTE OUTPUT FORMAT		
Column	**MODE Column Name**	**Description**
1	VERSION	Version number
2	MODEL	User provided text string designating model name
3	N_VALID	Number of valid data points
4	GRID_RES	User provided nominal grid resolution
5	DESC	User provided text string describing the verification task
6	FCST_LEAD	Forecast lead time in HHMMSS format
7	FCST_VALID	Forecast valid start time in YYYYMMDD_HHMMSS format
8	FCST_ACCUM	Forecast accumulation time in HHMMSS format
9	OBS_LEAD	Observation lead time in HHMMSS format; when field2 is actually an observation, this should be "000000"
10	OBS_VALID	Observation valid start time in YYYYMMDD_HHMMSS format
11	OBS_ACCUM	Observation accumulation time in HHMMSS format
12	FCST_RAD	Forecast convolution radius in grid squares
13	FCST_THR	Forecast convolution threshold
14	OBS_RAD	Observation convolution radius in grid squares
15	OBS_THR	Observation convolution threshold
16	FCST_VAR	Forecast variable
17	FCST_UNITS	Units for forecast variable
18	FCST_LEV	Forecast vertical level
19	OBS_VAR	Observation variable
20	OBS_UNITS	Units for observation variable
21	OBS_LEV	Observation vertical level
22	OBTYPE	User provided observation type
23	OBJECT_ID	Object numbered from 1 to the number of objects in each field
24	OBJECT_CAT	Object category indicating to which cluster object it belongs
25-26	CENTROID_X, _Y	Location of the centroid (in grid units)
27-28	CENTROID_LAT, _LON	Location of the centroid (in lat/lon degrees)
29	AXIS_ANG	Object axis angle (in degrees)
30	LENGTH	Length of the enclosing rectangle (in grid units)
31	WIDTH	Width of the enclosing rectangle (in grid units)
32	AREA	Object area (in grid squares)
33	AREA_THRESH	Area of the object containing data values in the raw field that meet the object definition threshold criteria (in grid squares)
34	CURVATURE	Radius of curvature of the object defined in terms of third order moments (in grid units)
35-36	CURVATURE_X, _Y	Center of curvature (in grid coordinates)
37	COMPLEXITY	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)
38-42	INTENSITY_10, _25, _50, _75, _90	10th, 25th, 50th, 75th, and 90th percentiles of intensity of the raw field within the object (various units)
43	INTENSITY_NN	The percentile of intensity chosen for use in the PERCENTILE_INTENSITY_RATIO column (variable units)

Table 15.4: Format of MODE object attribute output files, continued.

mode ASCII OBJECT ATTRIBUTE OUTPUT FORMAT		
Column	**MODE Column Name**	**Description**
44	INTENSITY_SUM	Sum of the intensities of the raw field within the object (variable units)
45	CENTROID_DIST	Distance between two objects centroids (in grid units)
46	BOUNDARY_DIST	Minimum distance between the boundaries of two objects (in grid units)
47	CONVEX_HULL _DIST	Minimum distance between the convex hulls of two objects (in grid units)
48	ANGLE_DIFF	Difference between the axis angles of two objects (in degrees)
49	ASPECT_DIFF	Absolute value of the difference between the aspect ratios of two objects (unitless)
50	AREA_RATIO	Ratio of the areas of two objects defined as the lesser of the two divided by the greater of the two (unitless)
51	INTERSECTION _AREA	Intersection area of two objects (in grid squares)
52	UNION_AREA	Union area of two objects (in grid squares)
53	SYMMETRIC_DIFF	Symmetric difference of two objects (in grid squares)
54	INTERSECTION _OVER_AREA	Ratio of intersection area to the lesser of the forecast and observation object areas (unitless)
55	CURVATURE _RATIO	Ratio of the curvature of two objects defined as the lesser of the two divided by the greater of the two (unitless)
56	COMPLEXITY _RATIO	Ratio of complexities of two objects defined as the lesser of the forecast complexity divided by the observation complexity or its reciprocal (unitless)
57	PERCENTILE _INTENSITY _RATIO	Ratio of the nth percentile (INTENSITY_NN column) of intensity of the two objects defined as the lesser of the forecast intensity divided by the observation intensity or its reciprocal (unitless)
58	INTEREST	Total interest value computed for a pair of simple objects (unitless)

NetCDF Output

The MODE tool creates a NetCDF output file containing the object fields that are defined. The NetCDF file contains gridded fields including indices for the simple forecast objects, indices for the simple observation objects, indices for the matched cluster forecast objects, and indices for the matched cluster observation objects. The NetCDF file also contains lat/lon and x/y data for the vertices of the polygons for the boundaries of the simple forecast and observation objects. The generation of this file can be disabled using the **nc_pairs_flag** configuration file option.

The dimensions and variables included in the mode NetCDF files are described in Tables 15.5, 15.6 and 15.7.

Table 15.5: NetCDF dimensions for MODE output.

mode NETCDF DIMENSIONS	
NetCDF Dimension	**Description**
lat	Dimension of the latitude (i.e. Number of grid points in the North-South direction)
lon	Dimension of the longitude (i.e. Number of grid points in the East-West direction)
fcst_thresh_length	Number of thresholds applied to the forecast
obs_thresh_length	Number of thresholds applied to the observations
fcst_simp	Number of simple forecast objects
fcst_simp_bdy	Number of points used to define the boundaries of all of the simple forecast objects
fcst_simp_hull	Number of points used to define the hull of all of the simple forecast objects
obs_simp	Number of simple observation objects
obs_simp_bdy	Number of points used to define the boundaries of all of the simple observation objects
obs_simp_hull	Number of points used to define the hull of all of the simple observation objects
fcst_clus	Number of forecast clusters
fcst_clus_hull	Number of points used to define the hull of all of the cluster forecast objects
obs_clus	Number of observed clusters
obs_clus_hull	Number of points used to define the hull of all of the cluster observation objects

Table 15.6: Variables contained in MODE NetCDF output.

mode NETCDF VARIABLES		
NetCDF Variable	**Dimension**	**Description**
lat	lat, lon	Latitude
lon	lat, lon	Longitude
fcst_raw	lat, lon	Forecast raw values
fcst_obj_raw	lat, lon	Forecast Object Raw Values
fcst_obj_id	lat, lon	Simple forecast object id number for each grid point
fcst_clus_id	lat, lon	Cluster forecast object id number for each grid point
obs_raw	lat, lon	Observation Raw Values
obs_obj_raw	lat, lon	Observation Object Raw Values
obs_obj_id	-	Simple observation object id number for each grid point
obs_clus_id	-	Cluster observation object id number for each grid point
fcst_conv_radius	-	Forecast convolution radius
obs_conv_radius	-	Observation convolution radius
fcst_conv _threshold	-	Forecast convolution threshold
obs_conv _threshold	-	Observation convolution threshold
n_fcst_simp	-	Number of simple forecast objects
n_obs_simp	-	Number of simple observation objects
n_clus		Number of cluster objects

Table 15.7: Variables contained in MODE NetCDF output - Simple Objects, continued from Table 15.6

mode NETCDF VARIABLES		
NetCDF Variable	**Dimension**	**Description**
fcst_simp_bdy_start	fcst_simp	Forecast Simple Boundary Starting Index
fcst_simp_bdy_npts	fcst_simp	Number of Forecast Simple Boundary Points
fcst_simp_bdy_lat	fcst_simp_bdy	Forecast Simple Boundary PoLatitude
fcst_simp_bdy_lon	fcst_simp_bdy	Forecast Simple Boundary PoLongitude
fcst_simp_bdy_x	fcst_simp_bdy	Forecast Simple Boundary PoX-Coordinate
fcst_simp_bdy_y	fcst_simp_bdy	Forecast Simple Boundary PoY-Coordinate
fcst_simp_hull_start	fcst_simp	Forecast Simple Convex Hull Starting Index
fcst_simp_hull_npts	fcst_simp	Number of Forecast Simple Convex Hull Points
fcst_simp_hull_lat	fcst_simp_hull	Forecast Simple Convex Hull Point Latitude
fcst_simp_hull_lon	fcst_simp_hull	Forecast Simple Convex Hull Point Longitude
fcst_simp_hull_x	fcst_simp_hull	Forecast Simple Convex Hull Po nt X-Coordinate
fcst_simp_hull_y	fcst_simp_hull	Forecast Simple Convex Hull Point Y-Coordinate
obs_simp_bdy_start	obs_simp	Observation Simple Boundary Starting Index
obs_simp_bdy_npts	obs_simp	Number of Observation Simple Boundary Points
obs_simp_bdy_lat	obs_simp_bdy	Observation Simple Boundary Point Latitude
obs_simp_bdy_lon	obs_simp_bdy	Observation Simple Boundary Point Longitude
obs_simp_bdy_x	obs_simp_bdy	Observation Simple Boundary Point X-Coordinate
obs_simp_bdy_y	obs_simp_bdy	Observation Simple Boundary Point Y-Coordinate
obs_simp_hull_start	obs_simp	Observation Simple Convex Hull Starting Index
obs_simp_hull_npts	obs_simp	Number of Observation Simple Convex Hull Points
obs_simp_hull_lat	obs_simp_hull	Observation Simple Convex Hull Point Latitude
obs_simp_hull_lon	obs_simp_hull	Observation Simple Convex Hull Point Longitude
obs_simp_hull_x	obs_simp_hull	Observation Simple Convex Hull Point X-Coordinate
obs_simp_hull_y	obs_simp_hull	Observation Simple Convex Hull Point Y-Coordinate

Table 15.8: Variables contained in MODE NetCDF output - Clustered Objects, continued from Table15.7

mode NETCDF VARIABLES		
NetCDF Variable	**Dimension**	**Description**
fcst_clus_hull_start	fcst_clus	Forecast Cluster Convex Hull Starting Index
fcst_clus_hull_npts	fcst_clus	Number of Forecast Cluster Convex Hull Points
fcst_clus_hull_lat	fcst_clus_hull	Forecast Cluster Convex Hull Point Latitude
fcst_clus_hull_lon	fcst_clus_hull	Forecast Cluster Convex Hull Point Longitude
fcst_clus_hull_x	fcst_clus_hull	Forecast Cluster Convex Hull Point X-Coordinate
fcst_clus_hull_y	fcst_clus_hull	Forecast Cluster Convex Hull Point Y-Coordinate
obs_clus_hull_start	obs_clus	Observation Cluster Convex Hull Starting Index
obs_clus_hull_npts	obs_clus	Number of Observation Cluster Convex Hull Points
obs_clus_hull_lat	obs_clus_hull	Observation Cluster Convex Hull Point Latitude
obs_clus_hull_lon	obs_clus_hull	Observation Cluster Convex Hull Point Longitude
obs_clus_hull_x	obs_clus_hull	Observation Cluster Convex Hull Point X-Coordinate
obs_clus_hull_y	obs_clus_hull	Observation Cluster Convex Hull Point Y-Coordinate

Postscript File

Lastly, the MODE tool creates a PostScript plot summarizing the features-based approach used in the verification. The PostScript plot is generated using internal libraries and does not depend on an external plotting package. The generation of this PostScript output can be disabled using the **ps_plot_flag** configuration file option.

The PostScript plot will contain 5 summary pages at a minimum, but the number of pages will depend on the merging options chosen. Additional pages will be created if merging is performed using the double thresholding or fuzzy engine merging techniques for the forecast and/or observation fields. Examples of the PostScript plots can be obtained by running the example cases provided with the MET tarball.

The first page of PostScript output contains a great deal of summary information. Six tiles of images provide thumbnail images of the raw fields, matched/merged object fields, and object index fields for the forecast and observation grids. In the matched/merged object fields, matching colors of objects across fields indicate that the corresponding objects match, while within a single field, black outlines indicate merging. Note that objects that are colored royal blue are unmatched. Along the bottom of the page, the criteria used for object definition and matching/merging are listed. Along the right side of the page, total interest values for pairs of simple objects are listed in sorted order. The numbers in this list correspond to the object indices shown in the object index plots.

The second and third pages of the PostScript output file display enlargements of the forecast and observation raw and object fields, respectively. The fourth page displays the forecast object with the outlines of the

observation objects overlaid, and vice versa. The fifth page contains summary information about the pairs of matched cluster objects.

If the double threshold merging or the fuzzy engine merging techniques have been applied, the output from those steps is summarized on additional pages.

Chapter 16

MODE-Analysis Tool

16.1 Introduction

Users may wish to summarize multiple ASCII files produced by MODE across many cases. The MODE output files contain many output columns making it very difficult to interpret the results by simply browsing the files. Furthermore, for particular applications some data fields in the MODE output files may not be of interest. The MODE-Analysis tool provide a simple way to compute basic summary statistics and filtering capabilities for these files. Users who are not proficient at writing scripts can use the tool directly, and even those using their own scripts can use this tool as a filter, to extract only the MODE output lines that are relevant for their application.

16.2 Scientific and statistical aspects

The MODE-Analysis tool operates in two modes, called "summary" and "bycase". In summary mode, the user specifies on the command line the MODE output columns of interest as well as filtering criteria that determine which input lines should be used. For example, a user may be interested in forecast object areas, but only if the object was matched, and only if the object centroid is inside a particular region. The summary statistics generated for each specified column of data are the minimum, maximum, mean, standard deviation, and the 10th, 25th, 50th, 75th and 90th percentiles. In addition, the user may specify a "dump"" file: the individual MODE lines used to produce the statistics will be written to this file. This option provides the user with a filtering capability. The dump file will consist only of lines that match the specified criteria.

The other option for operating the analysis tool is "bycase". Given initial and final values for forecast lead time, the tool will output, for each valid time in the interval, the matched area, unmatched area, and the number of forecast and observed objects that were matched or unmatched. For the areas, the user can specify forecast or observed objects, and also simple or cluster objects. A dump file may also be specified in this mode.

16.3 Practical information

The MODE-Analysis tool reads lines from MODE ASCII output files and applies filtering and computes basic statistics on the object attribute values. For each job type, filter parameters can be set to determine which MODE output lines are used. The following sections describe the **mode_analysis** usage statement, required arguments, and optional arguments.

16.3.1 mode_analysis usage

The usage statement for the MODE-Analysis tool is shown below:

```
Usage: mode_analysis
       -lookin path
       -summary | -bycase
       [-column name]
       [-dump_row filename]
       [-out filename]
       [-log file]
       [-v level]
       [-help]
       [MODE FILE LIST]
       [-config config_file] | [MODE LINE OPTIONS]
```

The MODE-Analysis tool has two required arguments and can accept several optional arguments.

Required arguments for mode_analysis:

1. The **-lookin path** specifies the name of a specific STAT file (any file ending in .stat) or the name of a directory where the Stat-Analysis tool will search for STAT files. This option may be used multiple times to specify multiple locations.

2. The MODE-Analysis tool can perform two basic types of jobs -summary or -bycase. Exactly one of these job types must be specified.

Specifying **-summary** will produce summary statistics for the MODE output column specified. For this job type, a column name (or column number) must be specified using the **-column** option. Column names are not case sensitive. The column names are the same as described in Section 15.3.3. More information about this option is provided in subsequent sections.

Specifying **-bycase** will produce a table of metrics for each case undergoing analysis. Any columns specified are ignored for this option.

Optional arguments for mode_analysis

3. The **mode_analysis** options are described in the following section. These are divided into sub-sections describing the analysis options and mode line options.

Analysis options

The general analysis options described below provide a way for the user to indicate configuration files to be used, where to write lines used to perform the analysis, and over which fields to generate statistics.

```
-config filename
```

This option gives the name of a configuration file to be read. The contents of the configuration file are described in Section 16.3.2.

```
-dump_row filename
```

Any MODE lines kept from the input files are written to `filename`.

```
-column column
```

Specifies which columns in the MODE output files to generate statistics for. Fields may be indicated by name (case insensitive) or column number (beginning at one). This option can be repeated to specify multiple columns.

MODE Command Line Options

MODE command line options are used to create filters that determine which of the MODE output lines that are read in, are kept. The MODE line options are numerous. They fall into seven categories: toggles, multiple set string options, multiple set integer options, integer max/min options, date/time max/min options, floating-point max/min options, and miscellaneous options. These options are described here.

Toggles

The MODE line options described in this section are shown in pairs. These toggles represent parameters that can have only one (or none) of two values. Any of these toggles may be left unspecified. However, if

neither option for each toggle is indicated, the analysis will produce results that combine data from both toggles. This may produce unintended results.

`-fcst | -obs`

This toggle indicates whether forecast or observed lines should be used for analysis.

`-single | -pair`

This toggle indicates whether single object or object pair lines should be used.

`-simple | -cluster`

This toggle indicates whether simple object or cluster object lines should be used.

`-matched | -unmatched`

This toggle indicates whether matched or unmatched object lines should be used.

Multiple-set string options

The following options set various string attributes. They can be set multiple times on the command line but must be separated by spaces. Each of these options must be indicated as a string. String values that include spaces may be used by enclosing the string in quotation marks.

`-model value`

This option specifies which model to use; value must be a string.

```
-fcst_thr value
-obs_thr  value
```

These two options specify thresholds for forecast and observation objects to be used in the analysis, respectively.

```
-fcst_var value
-obs_var  value
```

These options indicate the names of variables to be used in the analysis for forecast and observed fields.

```
-fcst_units value
-obs_units  value
```

These options indicate the units to be used in the analysis for forecast and observed fields.

```
-fcst_lev value
-obs_lev  value
```

These options indicate vertical levels for forecast and observed fields to be used in the analysis.

Multiple-set integer options

The following options set various integer attributes. They can be set multiple times on the command line but must be separated by spaces. Each of the following options may only be indicated as an integer.

```
-fcst_lead value
-obs_lead  value
```

These options are integers of the form HH[MMSS] specifying an (hour-minute-second) lead time.

```
-fcst_accum value
-obs_accum  value
```

These options are integers of the form HHMMSS specifying an (hour-minute-second) accumulation time.

```
-fcst_rad value
-obs_rad  value
```

These options indicate the convolution radius used for forecast or observed objects, respectively.

Integer max/min options

These options set limits on various integer attributes. Leaving a maximum value unset means no upper limit is imposed on the value of the attribute. The option works similarly for minimum values.

```
-area_min value
-area_max value
```

These options are used to indicate minimum/maximum values for the area attribute to be used in the analysis.

```
-area_filter_min value
-area_filter_max value
```

These options are used to indicate minimum/maximum values accepted for the area filter. The area filter refers to the number of non-zero values of the raw data found within the object.

```
-area_thresh_min value
-area_thresh_max value
```

These options are used to indicate minimum/maximum values accepted for the area thresh. The area thresh refers to the number of values of the raw data found within the object that meet the object definition threshold criteria used.

```
-intersection_area_min value
-intersection_area_max value
```

These options refer to the minimum/maximum values accepted for the intersection area attribute.

```
-union_area_min value
-union_area_max value
```

These options refer to the minimum/maximum union area values accepted for analysis.

```
-symmetric_diff_min value
-symmetric_diff_max value
```

These options refer to the minimum/maximum values for symmetric difference for objects to be used in the analysis.

Date/time max/min options

These options set limits on various date/time attributes. The values can be specified in one of three ways:

First, the options may be indicated by a string of the form YYYYMMDD_HHMMSS. This specifies a complete calendar date and time.

Second, they may be indicated by a string of the form YYYYMMDD_HH. Here, the minutes and seconds are assumed to be zero.

The third way of indicating date/time attributes is by a string of the form YYYYMMDD. Here, hours, minutes and seconds are assumed to be zero.

```
-fcst_valid_min YYYYMMDD[_HH[MMSS]]
-fcst_valid_max YYYYMMDD[_HH[MMSS]]
-obs_valid_min  YYYYMMDD[_HH[MMSS]]
-obs_valid_max  YYYYMMDD[_HH[MMSS]]
```

These options indicate minimum/maximum values for the forecast and observation valid times.

```
-fcst_init_min YYYYMMDD[_HH[MMSS]]
-fcst_init_max YYYYMMDD[_HH[MMSS]]
-obs_init_min  YYYYMMDD[_HH[MMSS]]
-obs_init_max  YYYYMMDD[_HH[MMSS]]
```

These two options indicate minimum/maximum values for forecast and observation initialization times.

Floating-point max/min options

Setting limits on various floating-point attributes. One may specify these as integers (i.e., without a decimal point), if desired. The following pairs of options indicate minimum and maximum values for each MODE attribute that can be described as a floating-point number. Please refer to Chapter 15.3.3 for a description of these attributes as needed.

```
-centroid_x_min value
-centroid_x_max value
```

```
-centroid_y_min value
-centroid_y_max value
```

```
-centroid_lat_min value
-centroid_lat_max value
```

```
-centroid_lon_min value
-centroid_lon_max value
```

```
-axis_ang_min value
-axis_ang_max value
```

```
-length_min value
-length_max value
```

```
-width_min value
-width_max value
```

```
-curvature_min value
-curvature_max value
```

```
-curvature_x_min value
-curvature_x_max value
```

```
-curvature_y_min value
-curvature_y_max value
```

```
-complexity_min value
-complexity_max value
```

```
-intensity_10_min value
-intensity_10_max value
```

```
-intensity_25_min value
-intensity_25_max value
```

```
-intensity_50_min value
-intensity_50_max value
```

```
-intensity_75_min value
-intensity_75_max value
```

```
-intensity_90_min value
-intensity_90_max value
```

```
-intensity_user_min value
-intensity_user_max value
```

```
-intensity_sum_min value
-intensity_sum_max value
```

```
-centroid_dist_min value
-centroid_dist_max value
```

```
-boundary_dist_min value
-boundary_dist_max value
```

```
-convex_hull_dist_min value
-convex_hull_dist_max value
```

```
-angle_diff_min value
-angle_diff_max value
```

```
-aspect_diff_min value
-aspect_diff_max value
```

```
-area_ratio_min value
-area_ratio_max value
```

```
-intersection_over_area_min value
-intersection_over_area_max value
```

```
-curvature_ratio_min value
-curvature_ratio_max value
```

```
-complexity_ratio_min value
-complexity_ratio_max value
```

```
-percentile_intensity_ratio_min value
-percentile_intensity_ratio_max value
```

```
-interest_min value
-interest_max value
```

Miscellaneous options

These options are used to indicate parameters that did not fall into any of the previous categories.

```
-mask_poly filename
```

This option indicates the name of a polygon mask file to be used for filtering. The format for these files is the same as that of the polyline files for the other MET tools.

```
-help
```

This option prints the usage message.

16.3.2 mode_analysis configuration file

To use the MODE-Analysis tool, the user must un-comment the options in the configuration file to apply them and comment out unwanted options. The options in the configuration file for the MODE-Analysis tools are the same as the MODE command line options described in Section16.3.1.

The parameters that are set in the configuration file either add to or override parameters that are set on the command line. For the "set string" and "set integer type" options enclosed in brackets, the values specified in the configuration file are added to any values set on the command line. For the "toggle" and "min/max type" options, the values specified in the configuration file override those set on the command line.

16.3.3 mode_analysis output

The output of the MODE-Analysis tool is a self-describing tabular format written to standard output. The length and contents of the table vary depending on whether **-summary** or **-bycase** is selected. The contents also change for **-summary** depending on the number of columns specified by the user.

Chapter 17

MODE Time Domain Tool

17.1 Introduction

17.1.1 Motivation

MODE Time Domain (MTD) is an extension of the MODE object-based approach to verification. In addition to incorporating spatial information, MTD utilizes the time dimension to get at temporal aspects of forecast verification. Since the two spatial dimensions of traditional meteorological forecasts are retained in addition to the time dimension, the method in inherently three dimensional. Given that, however, the overall methodology has deliberately been kept as similar as possible to that of traditional MODE.

A plot of some MTD precipitation objects is shown over the United States in Figure 17.1. The colors indicate longitude, with red in the east moving through the spectrum to blue in the west. Time increases vertically in this plot (and in most of the spacetime diagrams in this users' guide). A few things are worthy of note in this figure. First, the tendency of storm systems to move from west to east over time shows up clearly. Second, tracking of storm objects over time is easily done: if we want to know if a storm at one time is a later version of a storm at an earlier time, we need only see if they are part of the same 3D spacetime object. Lastly, storms splitting up or merging over time are handled easily by this method.

The 2D (or traditional) MODE approach to object-base verification enabled users to analyze forecasts in terms of location errors, intensity errors and shape, size and orientation errors. MTD retains all of that capability, and adds new classes of forecast errors involving time information: speed and direction errors, buildup and decay errors, and timing and duration errors. This opens up new ways of analyzing forecast quality.

In the past, many MET users have performed separate MODE runs at a series of forecast valid times and analyzed the resulting object attributes, matches and merges as functions of time in an effort to incorporate temporal information in assessments of forecast quality. MTD was developed as a way to address this need

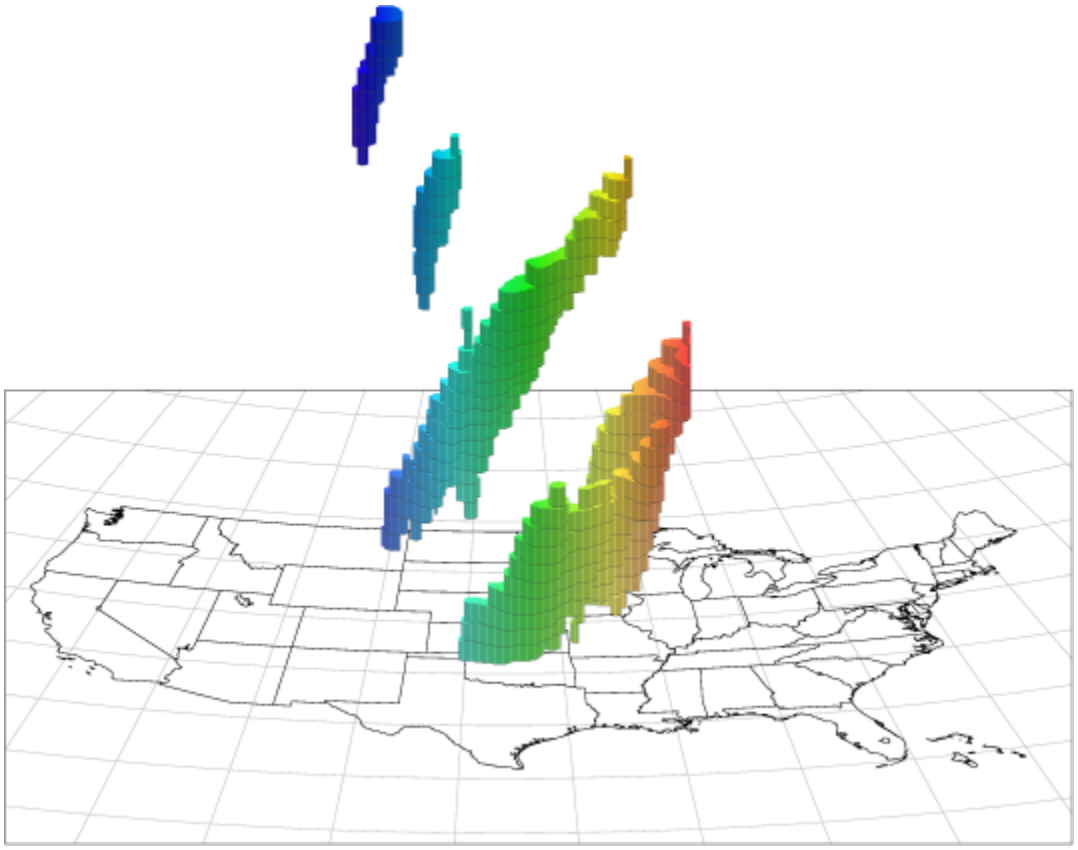

Figure 17.1: MTD Spacetime Objects

in a more systematic way. Most of the information obtained from such multiple coordinated MODE runs can be obtained more simply from MTD.

At first glance, the addition of a third dimension would seem to entail no difficulties other than increased memory and processing requirements to handle the three-dimensional datasets and objects, and that would indeed be largely true of an extension of MODE that used three spatial dimensions. In fact, the implementation of MTD entailed both conceptual difficulties (mostly due to the fact that there is no distance function in spacetime, so some MODE attributes, such as centroid distance, no longer even made sense), and engineering difficulties brought on by the need to redesign several core MODE algorithms for speed. It is planned that in the future some of these improved algorithms will be incorporated into MODE.

In this users' guide, we will assume that the reader has a basic familiarity with traditional MODE, its internal operation, (convolution thresholding, fuzzy logic matching and merging) and its output. We will not review these things here. Instead, we will point out differences in MTD from the way traditional MODE does things when they come up. This release is a beta version of MTD, intended mostly to encourage users to experiment with it and give us feedback and suggestions to be used in a more robust MTD release in the future.

17.2 Scientific and statistical aspects

17.2.1 Attributes

Object attributes are, for the most part, calculated in much the same way in MTD as they are in MODE, although the fact that one of the dimensions is non-spatial introduces a few quirks. Several of the object attributes that traditional MODE calculates assume that distances, angles and areas can be calculated in grid coordinates via the usual Euclidian/Cartesian methods. That is no longer the case in spacetime, since there is no distance function (more precisely, no *metric*) there. Given two points in this spacetime, say (x_1, y_1, t_1) and (x_2, y_2, t_2), there is no way to measure their separation with a single nonnegative number in a physically meaningful way. If all three of our dimensions were spatial, there would be no difficulties.

This means that some care must be taken both in determining how to generalize the calculation of a geometric attribute to three-dimensional spacetime, and also in interpreting the attributes even in the case where the generalization is straightforward.

17.2.2 Convolution

As in MODE, MTD applies a convolution filter to the raw data as a preliminary step in resolving the field into objects. The convolution step in MTD differs in several respects from that performed in MODE, however.

First, MTD typically reads in several planes of data for each data field—one plane for each time step, and there is really no limit to the number of time steps. So MTD is convolving much more data than it would

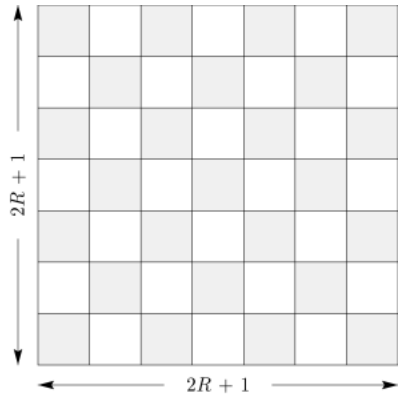

Figure 17.2: Convolution Region

be if it were simply analyzing a 2D data field. Secondly, MTD convolves in time as well as space, which again increases the amount of data needing to be processed. The net effect of all this is to greatly increase the time needed to perform the convolution step.

Because of this, the developers decided to make several changes in the way convolution was performed in MTD. Most of the differences come from the need to make the convolution step as fast as possible.

The most basic change is to use a square convolution filter rather than the circular one that MODE uses. The overall "size" of the filter is still determined by one parameter (denoted R, as in MODE), but this should not be thought of as a *radius*. Instead, the size of the square is $(2R+1) \times (2R+1)$, as shown in Figure 17.2.

Another change is that we do not allow any bad data in thee convolution square. in MODE, the user may specify what percentage of bad data in the convolution region is permissible, and MODE will rescale the value of the filter accordingly for each data point. For the sake of speed, MTD requires that there be no bad data in the convolution region. If any bad data exists in the region, the convolved value there is set to a bad data flag.

17.2.3 3D Single Attributes

MTD calculates several 3D attributes for single objects. The object could come from either the forecast field or the observed field.

A 3D spacetime **centroid** $(\overline{x}, \overline{y}, \overline{t})$ is calculated. There are no statistical overtones here. The number $\overline{x}$, for example, is just the average value of the x coordinate over the object.

The vector **velocity** (v_x, v_y) is obtained by fitting a line to an 3D object. The requirement for fitting the line is to minimize the sum of the squares of the *spatial* distances from each point of the object to the line be minimized. (We can't measure distances in spacetime but at each fixed time t we can measure purely spatial distances.) See Figure 17.3 for an illustration, where the solid line is the fitted axis, and the inclination of the axis from the vertical is a measure of object speed. Thus, from this velocity we get the **speed** and **direction** of movement of the object. As in MODE, where spatial separation is in units of the grid resolution, so here in

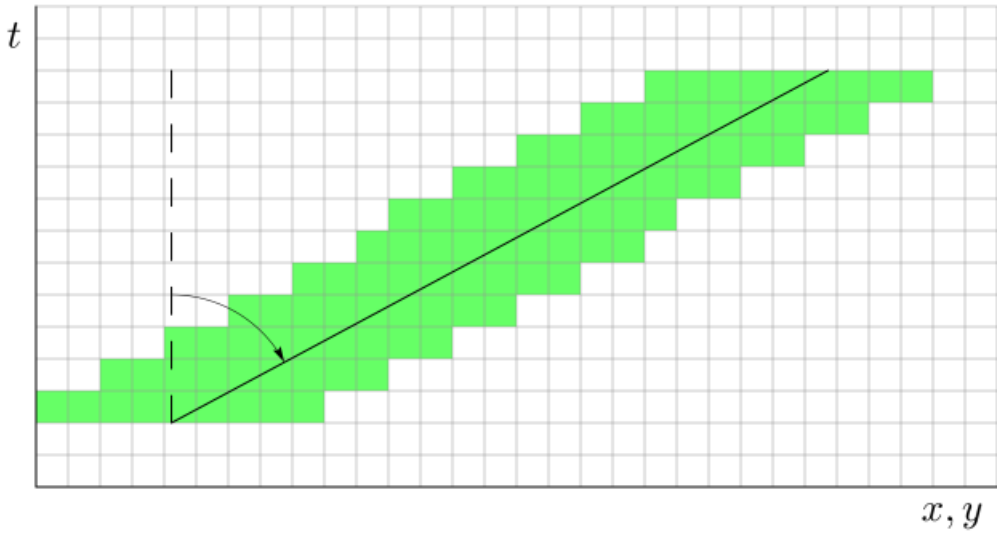

Figure 17.3: Velocity

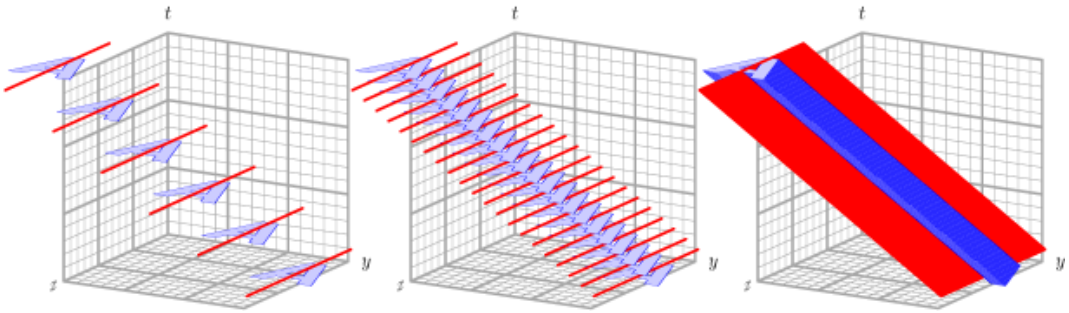

Figure 17.4: 3D axis

MTD the unit of length is the grid resolution, and the unit of time is whatever the time separation between the input files is. Speed and velocity are thus in grid units per time unit.

The spatial orientation of a object (what traditional MODE calls the **axis angle** of an object) is gotten by fitting a plane to an object. As with the case of velocity, our optimization criterion is that the sum of the squares of the spatial distances from each point of the object to the plane be minimized.

Figure 17.4 gives some idea of the reason for fitting a plane, rather than a line, as MODE does. On the left in the figure, we see an object (in blue shaped like an "A") at several time steps moving through the grid. For simplicity, the object is not rotating as it moves (though of course real objects can certainly do this). At each time step, the 2D MODE spatial axis of the object is indicated by the red line. In the center of the figure, we see the same thing, just with more time steps. And on the right, even more time steps. We see that the axis lines at each time step sweep out a plane in three dimensions, shown in red on the right. This plane is the same one that MTD would calculate for this 3D object to determine its spatial orientation, *i.e.*, axis angle. Indeed, for the special case of an object that is not moving at all, the MTD calculation of axis angle reduces to the same one that traditional MODE uses, as it should.

A simple integer count of the number of grid squares in an object for all of it's lifetime gives the **volume** of

the object. Remember that while we're working in three dimensions, one of the dimensions is non-spatial, so one should not attempt to convert this to a volume in, *e.g.*, km^3.

The **start time** and **end time** of an object are attributes as well. This is an integer telling which time step an object starts and ends at. These values are zero-based, so for example, if an object comes into existence at the 3^{rd} time step and lasts until the 9^{th} time step, then the start time and end time will be listed as 2 and 8, respectively. Note that this object has a lifetime of 7 time steps, not 6.

Centroid distance travelled is the total great circle distance, in kilometers, travelled by the 2D spatial centroid over the lifetime of the object. In other words, at each time t for which the 3D object exists, the set of points in the object also have that value of t will together form a 2D spatial object. That 2D object will have a spatial centroid, which will move around as t varies. This attribute represents this total 2D centroid movement over time.

Finally, MTD calculates several **intensity percentiles** of the raw data values inside each object. Not all of the the attributes are purely geometrical.

17.2.4 3D Pair Attributes

The next category of spatial attributes is for pairs of objects — one of the pair coming from the collection of forecast objects, the other coming from the observation objects.

Note: whenever a pair attribute is described below as a *delta*, that means it's a simple difference of two single-object attributes. The difference is always taken as "forecast minus observed".

The **spatial centroid distance** is the purely spatial part of the centroid separation of two objects. If one centroid is at $(\overline{x}_1, \overline{y}_1, \overline{t}_1)$ and the other is at $(\overline{x}_2, \overline{y}_2, \overline{t}_2)$ then the distance is calculated as

$$\sqrt{(\overline{x}_1 - \overline{x}_2)^2 + (\overline{y}_1 - \overline{y}_2)^2}$$

The **time centroid delta** is the difference between the time coordinates of the centroid. Since this is a simple difference, it can be either positive or negative.

The **axis difference** is smaller of the two angles that the two spatial axis planes make with each other. Figure 17.5 shows the idea. In the figure, the axis angle would be reported as angle α, not angle β.

Speed delta and **direction difference** are obtained from the velocity vectors of the two objects. Speed delta is the difference in the lengths of the vectors, and direction difference is the angle that the two vectors make with each other.

Volume ratio is volume of the forecast object divided by the volume of the observed object. Note that any 3D object must necessarily have a nonzero volume, so there's no chance of zeros in the denominator.

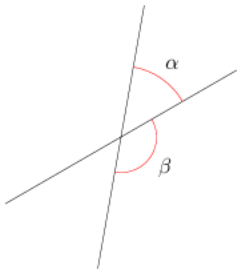

Figure 17.5: Axis Angle Difference

Start time delta and **end time delta** are the differences in the corresponding time steps associated with the two objects and are computed as "forecast minus obs".

Intersection volume measures the overlap of two objects. If the two objects do not overlap, then this will be zero.

Duration difference is the difference in the lifetimes of the two objects constituting the pair, in the sense of "forecast minus obs". For example, if the forecast object of the pair has a lifetime of 5 time steps, and the observed object has a lifetime of 3 time steps, then this attribute has the value 2. Note that we do not take absolute values of the difference, so this attribute can be positive, negative, or zero.

Finally, the **total interest** gives the result of the fuzzy-logic matching and merging calculation for this pair of objects. Note that this is provided only for simple objects, not for clusters.

17.2.5 2D Constant-Time Attributes

The final category of object attributes calculated by MTD are two-dimensional spatial attributes for horizontal (*i.e.*, constant-time) slices of a spacetime object. This is so that the behavior of these attributes over time can be examined. These 2D constant-time attributes are written out for both simple and cluster objects.

For example, in our earlier discussion relating to Figure 17.4, we mentioned that for simplicity, the object in the figure was not allowed to rotate as it moved. But what if the object (a hurricane, for example) *is* rotating over time? In that case, it's probably not meaningful to assign a single spatial orientation to the object over its entire lifetime. If we had a spatial axis angle at each time, however, then we could fit a model such as $\theta = \theta_0 + \omega t$ to the angles and test the goodness of fit.

For such reasons, having 2D spatial attributes (as in MODE) for each object at each time step can be useful. The list of the 2D attributes calculated is:

- Centroid (x, y)
- Centroid latitude and longitude
- Area
- Axis Angle

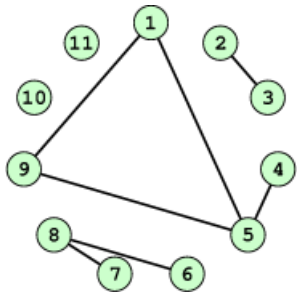

Figure 17.6: Basic Graph Example

17.2.6 Matching and Merging

Matching and merging operations in MTD are done in a simpler fashion than in MODE. In order to understand this operation, it is necessary to discuss some very basic notions of graph theory.

A **graph** is a finite set of **vertices** (also called **nodes**) and **edges**, with each edge connecting two vertices. Conceptually, it is enough for our purposes to think of vertices as points and edges as lines connecting them. See Figure 17.6 for an illustration. In the figure we see a collection of 11 nodes, indicated by the small circles, together with some edges indicated by straight line segments. A **path** is a sequence of vertices $(v_1, v_2, \ldots, v_n)$ such that for each $1 \leq i < n$ there is an edge connecting v_i to v_{i+1}. For example, in Figure 17.6, there is no edge connecting vertices #6 and #7, but there is a path connecting them. In illustrations, graph vertices are often labelled with identifying information, such as the numbers in Figure 17.6.

If we consider two distinct nodes in a graph to be related if there is a path connecting them, then it's easy to see that this defines an equivalence relation on the set of nodes, partitioning the graph into equivalence classes. Any node, such as #10 in Figure 17.6, that has no edges emanating from it is in a class by itself.

We have barely scratched the surface of the enormous subject of graph theory, but this will suffice for our purposes. How does MTD use graphs? Essentially the simple forecast and observed objects become nodes in a graph. Each pair of objects that have sufficiently high total interest (as determined by the fuzzy logic engine) generates an edge connecting the two corresponding nodes in the graph. The graph is then partitioned into equivalence classes using path connectivity (as explained above), and the resulting equivalence classes determine the matches and merges.

An example will hopefully make this clear. In parts (a) and (b) of Figure 17.7 we indicate the objects in the forecast and observed field for this simple example. We have used 2D rather than 3D objects in this example for simplicity. Also, to help distinguish the objects in each field, the forecast objects are labelled by numbers and the observed object by letters. Each forecast and each observed object become nodes in a graph as indicated in part (c) of the figure.

For the purposes of this example, suppose that the MTD fuzzy engine reports that observed simple object B and forecast simple object 4 together have a total interest higher than the total interest threshold specified in the config file. Also, observed simple object C and forecast simple object 4 have high enough interest to pass the threshold. Furthermore, forecast simple objects 2 and 3 both have sufficiently high interest when paired with observed simple object A.

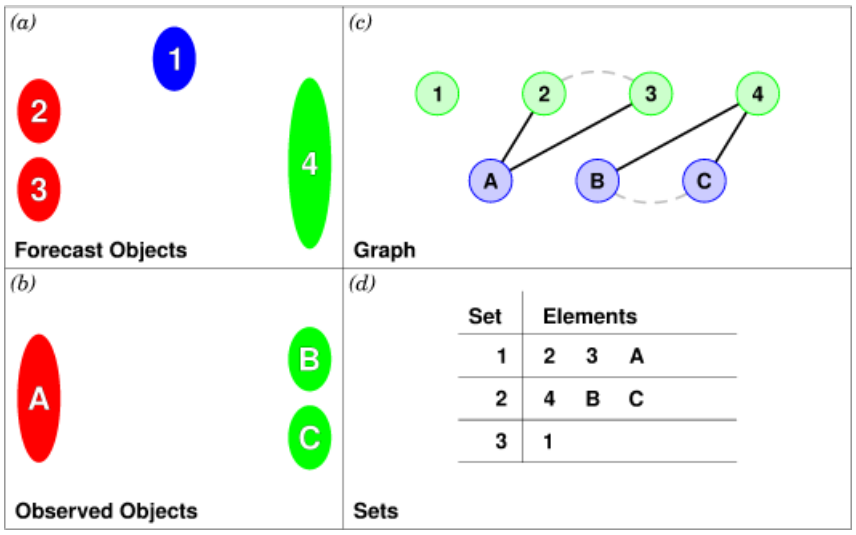

Figure 17.7: Match & Merge Example

These four pairings result in the 4 edges in the graph shown by the solid lines in part (c) of the figure. Partitioning this graph into equivalence classes results in the three sets indicated in part (d) of the figure. These three sets are the cluster objects determined by MTD. In this example, forecast objects 2 and 3 are merged into forecast cluster object #1 which is matched to observed cluster object #1, consisting of observed object A. (As in MODE, a cluster object may contain multiple simple objects, but may also consist of a single simple object.) Essentially, forecast simple objects 2 and 3 are merged because there is a path connecting them in the graph. This is indicated by the dashed line in the graph.

Continuing this example, forecast cluster object #2 (consisting only of forecast simple object 4) is matched to observed cluster object #2 (consisting of observed simple objects B and C). Again, the merging of observed simple objects is indicated by the dashed line in the graph.

Forecast cluster object #3 consists solely of forecast simple object 1. It is not matched to any observed cluster object. Alternatively, one may take the viewpoint that forecast simple object 1 ended up not participating in the matching and merging process; it is not merged with anything, it is not matched with anything. Essentially it represents a false alarm.

To summarize: Any forecast simple objects that find themselves in the same equivalence class are merged. Similarly, any observed objects in the same class are merged. Any forecast and observed objects in the same class are matched.

17.3 Practical information

17.3.1 MTD input

The formats for two-dimensional data files used as input to MTD are the same ones supported by most of the MET tools. Generally speaking, if MODE can use a forecast or observation data file as input, then that

file can also be used by MTD. The only difference is that while MODE takes only one forecast and one observed data file as input, MTD takes a series of files.

As shown in the next section, filenames for each time used must be given. Thus, for example, if MTD is being used for verification over a period of 24 hours, and the data file valid times are separated by one hour, then a total of 48 filenames must be specified on the MTD command line — 24 filenames for the forecast files, and 24 for the observation files. Further, the filenames must be given in order of increasing valid time. Many users will prefer to write scripts to automate this, rather than type in a lengthy command line by hand.

17.3.2 MTD usage

The usage statement for the MODE-TD tool is listed below: The command line switches may be given in any order.

```
Usage: mtd
       -fcst    file_1 ... file_n | file_list
       -obs     file_1 ... file_n | file_list
       -single  file_1 ... file_n | file_list
       -config  config_file
       [-outdir path]
       [-log    file]
       [-v      level]
```

The MODE-TD tool has three required arguments and can accept several optional arguments.

Required arguments for mtd

1. `-fcst file_list` gives a list of forecast 2D data files to be processed by MTD. The files should have equally-spaced intervals of valid time.

2. `-obs file_list` gives a list of observation 2D data files to be processed by MTD. As with the {\cb -fcst} option, the files should have equally-spaced intervals of valid time. This valid time spacing should be the same as for the forecast files.

3. `-config config_file` gives the path to a local configuration file that is specific to this particular run of MTD. The default MTD configuration file will be read first, followed by this one. Thus, only configuration options that are different from the default settings need be specified. Options set in this file will override any corresponding options set in the default configuration file.

Optional arguments for mtd

4. `-single file_list` may be used instead of `-fcst` and `-obs` to define objects in a single field.

5. `-log file` gives the name of a file where a log of this MTD run will be written. All output that appears on the screen during a MTD run will be duplicated in the log file.

6. `-v level` gives the verbosity level. As with the `-log` option described above, this option is present in most of the MET tools. Increasing this value causes more diagnostic output to be written to the screen (and also to the log file, if one has been specified).

7. `-outdir path` gives the name of the directory into which MTD will write its output files. If not specified, then MTD will write its output into the current directory.

An example of the mtd calling sequence is listed below:

```
mtd -fcst fcst_files/*.grb \
   -obs obs_files/*.grb \
   -config MTDConfig_default \
   -outdir out_dir/mtd \
   -v 1
```

In this example, the MODE-TD tool will read in a list of forecast GRIB file in the fcst_files directory and a similarly spaced observation GRIB files in the obs_files director. It uses a configuration file called MTDConfig_default and writes the output to out_dir/mtd directory.

17.3.3 MTD configuration file

The default configuration file for the MODE tool, MODEConfig_default, can be found in the installed share/met/config directory. Another version of the configuration file is provided in scripts/config. We encourage users to make a copy of the configuration files prior to modifying their contents.Most of the entries in the MTD configuration file should be familiar from the corresponding file for MODE. This initial beta release of MTD does not offer all the tunable options that MODE has accumulated over the years, however. In this section, we will not bother to repeat explanations of config file details that are exactly the same as those in MODE; we will only explain those elements that are different from MODE, and those that are unique to MTD.

```
model          = "WRF";
desc           = "NA";
obtype         = "ANALYS";
regrid         = { ... }
met_data_dir   = "MET_BASE";
output_prefix  = "";
version        = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
grid_res = 4;
fcst = {
   field = {
      name  = "APCP";
      level = "A03";
   }
   conv_time_window  = { beg = -1; end = 1; }
   conv_radius       = 60.0/grid_res; // in grid squares
   conv_thresh       = >=5.0;
}
obs = fcst;
total_interest_thresh = 0.7;
```

The configuration options listed above are common to many MODE and are described in Section 15.3.2.

The **conv_time_window** entry is a dictionary defining how much smoothing in time should be done. The **beg** and **end** entries are integers defining how many time steps should be used before and after the current time. The default setting of **beg = -1; end = 1;** uses one time step before and after. Setting them both to 0 effectively disables smoothing in time.

```
inten_perc_value = 99;
```

The **inten_perc_value** entry is an integer between 0 and 100 which specifies a requested intensity percentile value. By default, MTD writes 5 output columns for the 10th, 25th, 50th, 75th, and 90th percentile of object intensities. The percentile value specified here indicates which percentile should be written to the 6th output column.

```
min_volume = 2000;
```

The **min_volume** entry tell MTD to throw away objects whose "volume" (as described elsewhere in this chapter) is smaller than the given value. Spacetime objects whose volume is less than this will not participate in the matching and merging process, and no attribute information will be written to the ASCII output files. The default value is 10,000. If this seems rather large, consider the following example: Suppose the user is running MTD on a 600×400 grid, using 24 time steps. Then the volume of the whole data field is

$600 \times 400 \times 24 = 5{,}760{,}000$ cells. An object of volume 10,000 represents only $10{,}000/5{,}760{,}000 = 1/576$ of the total data field. Setting `min_volume` too small will typically produce a very large number of small objects, slowing down the MTD run and increasing the size of the output files.The configuration options listed above are common to many MODE and are described in Section 15.3.2.

```
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;
}
```

The **weight** entries listed above control how much weight is assigned to each pairwise attribute when computing a total interest value for object pairs. See Table 17.4 for a description of each weight option. When the total interest value is computed, the weighted sum is normalized by the sum of the weights listed above.

```
interest_function = {
   space_centroid_dist = ( ... );
   time_centroid_delta = ( ... );
   speed_delta         = ( ... );
   direction_diff      = ( ... );
   volume_ratio        = ( ... );
   axis_angle_diff     = ( ... );
   start_time_delta    = ( ... );
   end_time_delta      = ( ... );
};
```

The **interest_function** entries listed above control how much weight is assigned to each pairwise attribute when computing a total interest value for object pairs. See Table 17.4 for a description of each weight option. The interest functions may be defined as a piecewise linear function or as an algebraic expression. A piecewise linear function is defined by specifying the corner points of its graph. An algebraic function may be defined in terms of several built-in mathematical functions. See Section 15.2 for how interest values are used by the fuzzy logic engine. By default, many of these functions are defined in terms of the previously defined **grid_res** entry.

```
nc_output = {
   latlon     = true;
   raw        = true;
   object_id  = true;
   cluster_id = true;
};
```

The **nc_output** dictionary contains a collection of boolean flags controlling which fields are written to the NetCDF output file. **latlon** controls the output of a pair of 2D fields giving the latitude and longitude of each grid point. The **raw** entry controls the output of the raw input data for the MTD run. These will be 3D fields, one for the forecast data and one for the observation data. Finally, the **object_id** and **cluster_id** flags control the output of the object numbers and cluster numbers for the objects. This is similar to MODE.

```
txt_output = {
   attributes_2d = true;
   attributes_3d = true;
};
```

The **txt_output** dictionary also contains a collection of boolean flags, in this case controlling the output of ASCII attribute files. The **attributes_2d** flag controls the output of the 2D object attributes for constant-time slices of 3D objects, while the **attributes_3d** flag controls the output of single and pair 3D spacetime object attributes.

17.3.4 mtd output

MTD creates several output files after each run in ASCII and NetCDF formats. There are text files giving 2D and 3D attributes of spacetime objects and information on matches and merges, as well as a NetCDF file giving the objects themselves, in case any further or specialized analysis of the objects needs to be done.

MODE, along with several other of the MET tools (`wavelet_stat` for example, and a few others), provides PostScript-based graphics output to help visualize the output. Unfortunately, no similar graphics capabilities are provided with MTD, mainly because of the complexity of producing 3D plots. This should not discourage the user from making their own plots, however. There is enough information in the various output files created by MTD to make a wide variety of plots. Highly motivated users who write their own plotting scripts are encouraged to submit them to the user-contributed code area of the MET website. Due credit will be given, and others will benefit from their creations.

ASCII output

Five ASCII output files are created:

- Single attributes for 3D simple objects
- Single attributes for 3D cluster objects
- Pair attributes for 3D simple objects
- Pair attributes for 3D cluster objects
- 2D spatial attributes for single simple objects for each time index of their existence.

Each ASCII file is laid out in tabular format, with the first line consisting of text strings giving names for each column. The first 15 columns of each file are identical, and give information on timestamps, model names, and the convolution radius and threshold used for the forecast and observation input data.

These columns are explained in Table 17.1. Each file contains additional columns that come after these. Columns for 2D constant-time attributes are shown in Table 17.2. Columns for 3D single and pair attributes are shown in Tables 17.3 and 17.4 respectively.

The contents of the OBJECT_ID and OBJECT_CAT columns identify the objects using the same logic as the MODE tool. In these columns, the F and O prefixes are used to indicate simple forecast and observation objects, respectively. Similarly, the CF and CO prefixes indicate cluster forecast and observation objects, respectively. Each prefix is followed by a 3-digit number, using leading zeros, to indicate the object number (as in `F001`, `O001`, `CF001`, or `CO000`). Pairs of objects are indicated by listing the forecast object information followed by the observation object information, separated by an underscore (as in `F001_O001` or `CF001_CO001`). The OBJECT_ID column indicates the single object or pair of objects being described in that line. The OBJECT_CAT column indicates the cluster or pair of clusters to which these object(s) belong. A simple object that is not part of a cluster is assigned a cluster number of zero (as in `CF000` or `CO000`). When pairs of objects belong to the same matching cluster, the OBJECT_CAT column indicates the matching cluster number (as in `CF001_CO001`). When they do not, the OBJECT_CAT column is set to `CF000_CO000`.

NetCDF File

MTD writes a NetCDF file containing various types of information as specified in the configuration file. The possible output data are:

- **Latitude** and **longitude** of all the points in the 2D grid. Useful for geolocating points or regions given by grid coordinates.
- **Raw data** from the input data files. This can be useful if the input data were grib format, since NetCDF is often easier to read.
- **Object ID** numbers, giving for each grid point the number of the simple object (if any) that covers that point. These numbers are one-based. A value of zero means that this point is not part of any object.
- **Cluster ID** numbers. As above, only for cluster objects rather than simple objects.

Table 17.1: Text Header Columns

HEADER		
Column	**Name**	**Description**
1	VERSION	Version number
2	MODEL	User provided text string giving model name
3	DESC	User provided text string describing the verification task
4	FCST_LEAD	Forecast lead time in HHMMSS format
5	FCST_VALID	Forecast valid time in YYYYMMDD_HHMMSS format
6	OBS_LEAD	Observation lead time in HHMMSS format
7	OBS_VALID	Observation valid time in YYYYMMDD_HHMMSS format
8	T_DELTA	Time separation between input data files in HHMMSS format
9	FCST_T_BEG	Forecast time convolution begin offset
10	FCST_T_END	Forecast time convolution end offset
11	FCST_RAD	Forecast convolution radius in grid units
12	FCST_THR	Forecast convolution threshold
13	OBS_T_BEG	Observation time convolution begin offset
14	OBS_T_END	Observation time convolution end offset
15	OBS_RAD	Observation convolution radius in grid units
16	OBS_THR	Observation convolution threshold
17	FCST_VAR	Forecast variable
18	FCST_UNITS	Units for forecast variable
19	FCST_LEV	Forecast vertical level
20	OBS_VAR	Observation variable
21	OBS_UNITS	Units for observation variable
22	OBS_LEV	Observation vertical level

Table 17.2: 2D Attribute

2D Attribute Columns		
Column	**Name**	**Description**
23	OBJECT_ID	Object number
24	OBJECT_CAT	Object category
25	TIME_INDEX	Time index of slice
26	AREA	2D cross-sectional area
27	CENTROID_X	x coordinate of centroid
28	CENTROID_Y	y coordinate of centroid
29	CENTROID_LAT	Latitude of centroid
30	CENTROID_LON	Longitude of centroid
31	AXIS_ANG	Angle that the axis makes with the grid x direction
32	INTENSITY_10	10^{th} percentile intensity in time slice
33	INTENSITY_25	25^{th} percentile intensity in time slice
34	INTENSITY_50	50^{th} percentile intensity in time slice
35	INTENSITY_75	75^{th} percentile intensity in time slice
36	INTENSITY_90	90^{th} percentile intensity in time slice
37	INTENSITY_*	User-specified percentile intensity in time slice

Table 17.3: 3D Single Attribute

3D Single Attribute Columns		
Column	**Name**	**Description**
23	OBJECT_ID	Object number
24	OBJECT_CAT	Object category
25	CENTROID_X	x coordinate of centroid
26	CENTROID_Y	y coordinate of centroid
27	CENTROID_T	t coordinate of centroid
28	CENTROID_LAT	Latitude of centroid
29	CENTROID_LON	Longitude of centroid
30	X_DOT	x component of object velocity
31	Y_DOT	y component of object velocity
32	AXIS_ANG	Angle that the axis plane of an object makes with the grid x direction
33	VOLUME	Integer count of the number of 3D "cells" in an object
34	START_TIME	Object start time
35	END_TIME	Object end time
36	CDIST_TRAVELLED	Total great circle distance travelled by the 2D spatial centroid over the lifetime of the 3D object
37	INTENSITY_10	10^{th} percentile intensity inside object
38	INTENSITY_25	25^{th} percentile intensity inside object
39	INTENSITY_50	50^{th} percentile intensity inside object
40	INTENSITY_75	75^{th} percentile intensity inside object
41	INTENSITY_90	90^{th} percentile intensity inside object
42	INTENSITY_*	User-specified percentile intensity inside object

Table 17.4: 3D Pair Attribute

3D Pair Attribute Columns		
Column	**Name**	**Description**
23	OBJECT_ID	Object number
24	OBJECT_CAT	Object category
25	SPACE_CENTROID_DIST	Spatial distance between (x, y) coordinates of object spacetime centroid
26	TIME_CENTROID_DELTA	Difference in t index of object spacetime centroid
27	AXIS_DIFF	Difference in spatial axis plane angles
28	SPEED_DELTA	Difference in object speeds
29	DIRECTION_DIFF	Difference in object direction of movement
30	VOLUME_RATIO	Ratio of object volumes
31	START_TIME_DELTA	Difference in object starting time steps
32	END_TIME_DELTA	Difference in object ending time steps
33	INTERSECTION_VOLUME	"Volume" of object intersection
34	DURATION_DIFF	Dfference in the lifetimes of the two objects
35	INTEREST	Total interest for this object pair

Chapter 18

MET-TC Overview

18.1 Introduction

The purpose of this User's Guide is to provide basic information to the users of the Model Evaluation Tools - Tropical Cyclone (MET-TC) to enable users to apply MET-TC to their tropical cyclone datasets and evaluation studies. MET-TC is intended for use with model forecasts run through a vortex tracking software or with operational model forecasts in Automated Tropical Cyclone Forecast (ATCF) file format.

The following chapters provide an overview of MET-TC and its components, as well as basic information on the software build. The required input, including file format and the MET-TC are discussed followed by a description of the TC-Dland tool, TC-Pairs, and TC-Stat tools. Each chapter cover the input and output, and practical usage including a description of the configuration files. This is followed by a short overview of graphical utilities available within the MET-TC release.

18.2 MET-TC components

The MET tools used in the verification of Tropical Cyclones are referred to as MET-TC. These tools are shown across the bottom of the flowchart in Figure 1.1. The MET-TC tools are described in more detail in later chapters.

The TC-Dland tool is used to generate a gridded file that determines the location of coastlines and islands, and is used as input to the TC-Pairs tool to determine the distance from land of a particular track point. The TC-Pairs tool matches pairs of input model data and BEST track (or any reference forecast) and calculates position errors. The TC-Stat tool uses the TC-Pairs output to perform filter and summary jobs over the matched pair dataset. The TC-Gen tool performs a categorical analysis for tropical cyclone genesis forecasts. The TC-RMW tool performs a coordinate transformation of gridded model data, centered on the storm's location. The RMW-Analysis tool aggregates TC-RMW output across multiple cases.

18.3 Input data format

This chapter discusses the input and output file formats expected and produced by MET-TC. When discussing the input data, it is expected that users have run model output through vortex tracking software in order to obtain position and intensity information in Automated Tropical Cyclone Forecasting System (ATCF) file format. Best track and aids files in Automated Tropical Cyclone Forecasting System (ATCF) format (hereafter referred to as ATCF format) are necessary for model data input into the TC-Pairs tool. The ATCF format was first developed at the Naval Oceanographic and Atmospheric Research Laboratory (NRL), and is currently used for the National Hurricane Center (NHC) operations. ATCF format must be adhered to in order for the MET-TC tools to properly parse the input data.

The ATCF file format includes a section with common fields:

BASIN, CY, YYYYMMDDHH, TECHNUM/MIN, TECH, TAU, LatN/S, LonE/W, VMAX, MSLP, TY, RAD, WINDCODE, RAD1, RAD2, RAD3, RAD4, POUTER, ROUTER, RMW, GUSTS, EYE, SUBREGION, MAXSEAS, INITIALS, DIR, SPEED, STORMNAME, DEPTH, SEAS, SEASCODE, SEAS1, SEAS2, SEAS3, SEAS4

BASIN: basin

CY: annual cyclone number: 1 - 99

YYYYMMDDHH: Warning Date-Time-Group.

TECHNUM/MIN: objective technique sorting number, minutes for best track: 00 - 99

TECH: acronym for each objective technique or CARQ or WRNG, BEST for best track

TAU: forecast period: -24 through 240 hours, 0 for best-track

LatN/S: Latitude for the date time group (DTG)

LonE/W: Longitude for the DTG

VMAX: Maximum sustained wind speed in knots

MSLP: Minimum sea level pressure, 850 - 1050 mb.

TY: Highest level of tropical cyclone development

RAD: Wind intensity for the radii defined in this record: 34, 50 or 64 kt.

WINDCODE: Radius code

RAD1: If full circle, radius of specified wind intensity, or radius of first quadrant wind intensity

RAD2: If full circle this field not used, or radius of 2nd quadrant wind intensity

RAD3: If full circle this field not used, or radius of 3rd quadrant wind intensity

RAD4: If full circle this field not used, or radius of 4th quadrant wind intensity

POUTER: pressure in millibars of the last closed isobar

ROUTER: radius of the last closed isobar

RMW: radius of max winds

GUSTS: gusts

EYE: eye diameter

SUBREGION: subregion

MAXSEAS: max seas

INITIALS: Forecaster's initials

DIR: storm direction

SPEED: storm speed

STORMNAME: literal storm name, number, NONAME or INVEST, or TCcyx

DEPTH: system depth

SEAS: Wave height for radii defined in SEAS1 - SEAS4

SEASCODE - Radius code

SEAS1: first quadrant seas radius as defined by SEASCODE

SEAS2: second quadrant seas radius as defined by SEASCODE

SEAS3: third quadrant seas radius as defined by SEASCODE

SEAS4: fourth quadrant seas radius as defined by SEASCODE

Of the above common fields in the ATCF file format, MET-TC requires the input file has the first 8 comma-separated columns present. Although all 8 columns must exist, valid data in each field is not required. In order to ensure proper matching, unique data in the BASIN, CY, YYYYMMDDHH, and TAU fields should be present.

The TC-Pairs tool expects two input data sources in order to generate matched pairs and subsequent error statistics. The expected input for MET-TC is an ATCF format file from model output, or the operational aids files with the operational model output for the 'adeck' and the NHC best track analysis (BEST) for the

'bdeck'. The BEST is a subjectively smoothed representation of the storm's location and intensity over its lifetime. The track and intensity values are based on a retrospective assessment of all available observations of the storm.

The BEST is in ATCF file format and contains all the above listed common fields. Given the reference dataset is expected in ATCF file format, any second ATCF format file from model output or operational model output from the NHC aids files can be supplied as well. The expected use of the TC-Pairs tool is to generate matched pairs between model output and the BEST. Note that some of the columns in the TC-Pairs output are populated based on the BEST information (e.g. storm category), therefore use of a different baseline many reduce the available filtering options.

All operational model aids and the BEST can be obtained from the NHC ftp server: `ftp://ftp.nhc.noaa.gov/atcf/archive/`

For more detailed information on the ATCF format description and specifications see: `http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abdeck.txt`

In order to adhere to ATCF file format, model data must be run through a vortex tracking algorithm prior to becoming input for MET-TC. Many vortex tracking algorithms have been developed in order to obtain basic position, maximum wind, and minimum sea level pressure information from a model forecasts. One vortex tracking algorithm that is supported and freely available is the GFDL vortex tracker. Refer to `http://www.dtcenter.org/HurrWRF/users/downloads/index.php` for more information on the GFDL vortex tracker package.

18.4 Output data format

The MET package produces output in four basic file formats: STAT files, ASCII files, NetCDF files, and Postscript plots. The MET-TC tool produces output in TCSTAT, which stands for Tropical Cyclone - STAT. This output format consists of tabular ASCII data that can be easily read by many analysis tools and software packages, making the output from MET-TC very versatile. Like STAT, TCSTAT is a specialized ASCII format containing one record on each line. Currently, the only line type available in MET-TC is TCMPR (Tropical Cyclone Matched Pairs). As more line types are included in future releases, all line types will be included in a single TCSTAT file. MET-TC also outputs a NetCDF format file in the TC-Dland tool, as input to the TC-Pairs tool.

Chapter 19

TC-Dland Tool

19.1 Introduction

Many filtering criteria within the MET-TC tools depend on the distinction between when a storm is over land or water. The TC-dland tool was developed to aid in quickly parsing data for filter jobs that only verify over water, threshold verification based on distance to land, and exclusion of forecasts outside a specified time window of landfall. For each grid point in the user-specified grid, it computes the great circle arc distance to the nearest coast line. Great circle arc distances are more accurate but take considerably longer to compute than a simple Euclidean distance. Grid points over land have distances greater than zero while point over land have distances less than zero.

While the TC-dland tool is available to be run, most users will find the pre-computed distance to land files distributed with the release sufficient. Therefore, the typical user will not actually need to run this tool.

19.2 Input/output format

The input for the TC-dland tool is a file containing the longitude (degrees east) and latitude (degrees north) of all the coastlines and islands considered to be a significant landmass. The default input is to use all three land data files (aland.dat, shland.dat, wland.dat) found in the installed **share/met/tc_data/** directory. The use of all three files produces a global land data file. The aland.dat file contains the longitude and latitude distinctions used by NHC for the Atlantic and eastern North Pacific basins, the shland.dat contains longitude and latitude distinctions for the Southern Hemisphere (south Pacific and South Indian Ocean), and the wland.dat contains the remainder of the Northern Hemisphere (western North Pacific and North Indian Ocean). Users may supply their own input file in order to refine the definition of coastlines and a significant landmass.

The output file from TC-dland is a NetCDF format file containing a gridded field representing the distance to the nearest coastline or island, as specified in the input file. This file is used in the TC-Pairs tool to

compute the distance from land for each track point in the adeck and bdeck. As noted in chapter 1.3, pre-computed distance to land (NetCDF output from TC-dland) files are available in the release. In the installed **share/met/tc_data** directory:

dland_nw_hem_tenth_degree.nc: TC-dland output from aland.dat using a 1/10th degree grid

dland_global_tenth_degree.nc: TC-dland output from all three land data files (global coverage) using a 1/10th degree grid.

19.3 Practical information

This section briefly describes how to run tc_dland. The default grid is set to 1/10th degree NW hemisphere grid.

19.3.1 tc_dland usage

```
Usage: tc_dland
       out_file
       [-grid spec]
       [-noll]
       [-land file]
       [-log file]
       [-v level]
       [-compress level]
```

tc_dland has one required arguments and accepts several optional ones.

Required arguments for tc_dland

1. The **out_file** argument indicates indicates the NetCDF output file containing the computed distances to land.

Optional arguments for tc_dland

2. The **-grid spec** argument overrides the default grid (1/10th NH grid). Spec = **lat_ll lon_ll delta_lat delta_lon n_lat n_lon**

3. The **-noll** argument skips writing the lon/lat variables in the output NetCDF file to reduce the file size.

4. The **-land file** argument overwrites the default land data files (aland.dat, shland.dat, and wland.dat).

5. The **-log file** argument outputs log messages to the specified file.

6. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

7. The **-compress level** option specifies the desired level of compression (deflate level) for NetCDF variables. The valid level is between 0 and 9. Setting the compression level to 0 will make no compression for the NetCDF output. Lower number is for fast compression and higher number is for better compression.

Chapter 20

TC-Pairs Tool

20.1 Introduction

The TC-Pairs tool provides verification for tropical cyclone forecasts in ATCF file format. It matches an ATCF format tropical cyclone (TC) forecast with a second ATCF format reference TC dataset (most commonly the Best Track analysis). The TC-Pairs tool processes both track and intensity adeck data and probabilistic edeck data. The adeck matched pairs contain position errors, as well as wind, sea level pressure, and distance to land values for each TC dataset. The edeck matched pairs contain probabilistic forecast values and the verifying observation values. The pair generation can be subset based on user-defined filtering criteria. Practical aspects of the TC-Pairs tool are described in Section 20.2.

20.2 Practical information

This section describes how to configure and run the TC-Pairs tool. The TC-Pairs tool is used to match a tropical cyclone model forecast to a corresponding reference dataset. Both tropical cyclone forecast/reference data must be in ATCF format. Output from the TC-dland tool (NetCDF gridded distance file) is also a required input for the TC-Pairs tool. It is recommended to run **tc_pairs** on a storm-by-storm basis, rather than over multiple storms or seasons to avoid memory issues.

20.2.1 tc_pairs usage

The usage statement for tc_pairs is shown below:

```
Usage: tc_pairs
       -adeck source and/or -edeck source
```

```
-bdeck source
-config file
[-out base]
[-log file]
[-v level]
```

tc_pairs has required arguments and can accept several optional arguments.

Required arguments for tc_pairs

1. The **-adeck source** argument indicates the adeck ATCF format data source containing tropical cyclone model forecast (output from tracker) data to be verified. It specifies the name of an ATCF format file or top-level directory containing ATCF format files ending in ".dat" to be processed. The **-adeck** or **-edeck** option must be used at least once.

2. The **-edeck source** argument indicates the edeck ATCF format data source containing probabilistic track data to be verified. It specifies the name of an ATCF format file or top-level directory containing ATCF format files ending in ".dat" to be processed. The **-adeck** or **-edeck** option must be used at least once.

3. The **-bdeck source** argument indicates the ATCF format data source containing the tropical cyclone reference dataset to be used for verifying the adeck source. It specifies the name of an ATCF format file or top-level directory containing ATCF format files ending in ".dat" to be processed. This source is expected to be the NHC Best Track Analysis, but could also be any ATCF format reference.

4. The **-config file** argument indicates the name of the configuration file to be used. The contents of the configuration file are discussed below.

Optional arguments for tc_pairs

5. The **-out base** argument indicates the path of the output file base. This argument overrides the default output file base (./out_tcmpr)

6. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

7. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

MET version 6.0 supports only the rapid intensification (**RI**) edeck probability type but support for additional edeck probability types will be added in future releases. At least one **-adeck** or **-edeck** option must be specified. The **-adeck**, **-edeck**, and **-bdeck** options may optionally be followed with **suffix=string** to

append that string to all model names found within that data source. This option may be useful when processing track data from two different sources which reuse the same model names.

An example of the tc_pairs calling sequence is shown below:

```
tc_pairs -adeck aal092010.dat -bdeck bal092010.dat -config TCPairsConfig
```

In this example, the TC-Pairs tool matches the model track (aal092010.dat) and the best track analysis (bal092010.dat) for the 9th Atlantic Basin storm in 2010. The track matching and subsequent error information is generated with configuration options specified in the **TCPairsConfig** file.

The TC-Pairs tool implements the following logic:

- Parse the adeck, edeck, and bdeck data files and store them as track objects.
- Apply configuration file settings to filter the adeck, edeck, and bdeck track data down to a subset of interest.
- Apply configuration file settings to derive additional adeck track data, such as interpolated tracks, consensus tracks, time-lagged tracks, and statistical track and intensity models.
- For each adeck track that was parsed or derived, search for a matching bdeck track with the same basin and cyclone number and overlapping valid times. If not matching against the BEST track, also ensure that the model initialization times match.
- For each adeck/bdeck track pair, match up their track points in time, lookup distances to land, compute track location errors, and write an output TCMPR line for each track point.
- For each set of edeck probabilities that were parsed, search for a matching bdeck track.
- For each edeck/bdeck pair, write paired edeck probabilities and matching bdeck values to output PROBRIRW lines.

20.2.2 tc_pairs configuration file

The default configuration file for the TC-Pairs tool named 'TCPairsConfig_default' can be found in the installed **share/met/config/ directory**. It is encouraged for users to copy these default files before modifying their contents. The contents of the configuration file are described in the subsections below.

The contents of the tc_pairs configuration file are described below.

```
storm_id     = [];
basin        = [];
cyclone      = [];
storm_name   = [];
init_beg     = "";
init_end     = "";
init_inc     = [];
init_exc     = [];
valid_beg    = "";
valid_end    = "";
init_hour    = [];
init_mask    = [];
lead_req     = [];
valid_mask   = [];
match_points = TRUE;
version      = "VN.N";
```

The configuration options listed above are common to multiple MET tools and are described in Section 3.5.2.

```
model = [ "DSHP", "LGEM", "HWRF" ];
```

The **model** variable contains a list of comma-separated models to be used. The models are identified with an ATCF ID (normally four unique characters). This model identifier should match the model column in the ATCF format input file. An empty list indicates that all models in the input file(s) will be processed.

```
check_dup = FALSE;
```

The **check_dup** flag expects either TRUE and FALSE, indicating whether the code should check for duplicate ATCF lines when building tracks. Setting **check_dup** to TRUE will check for duplicated lines, and produce output information regarding the duplicate. The duplicated ATCF line will not be processed in the tc_pairs output. Setting **check_dup** to FALSE, will still exclude tracks that decrease with time, and will overwrite repeated lines, but specific duplicate log information will not be output. Setting **check_dup** to FALSE will make parsing the track quicker.

```
interp12 = NONE;
```

The **interp12** flag expects the entry NONE, FILL, or REPLACE, indicating whether special processing should be performed for interpolated forecasts. The NONE option indicates no changes are made to the interpolated forecasts. The FILL and REPLACE (default) options determine when the 12-hour interpolated forecast (normally indicated with a "2" or "3" at the end of the ATCF ID) will be renamed with the 6-hour interpolated ATCF ID (normally indicated with the letter "I" at the end of the ATCF ID). The FILL option renames the 12-hour interpolated forecasts with the 6-hour interpolated forecast ATCF ID only when the 6-hour interpolated forecasts is missing (in the case of a 6-hour interpolated forecast which only occurs every 12-hours (e.g. EMXI, EGRI), the 6-hour interpolated forecasts will be "filled in" with the 12-hour interpolated forecasts in order to provide a record every 6-hours). The REPLACE option renames all 12-hour interpolated forecasts with the 6-hour interpolated forecasts ATCF ID regardless of whether the 6-hour interpolated forecast exists. The original 12-hour ATCF ID will also be retained in the output file (all modified ATCF entries will appear at the end of the TC-Pairs output file). This functionality expects both the 12-hour and 6-hour early (interpolated) ATCF IDs are listed in the model field.

```
consensus = [
   {
      name     = "CON1";
      members  = [ "MOD1", "MOD2", "MOD3" ];
      required = [   true,  false, false  ];
      min_req  = 2;
   }
];
```

The **consensus** field allows the user to generate a user-defined consensus forecasts from any number of models. All models used in the consensus forecast need to be included in the **model** field (1st entry in TCPairsConfig_default). The name field is the desired consensus model name. The **members** field is a comma-separated list of model IDs that make up the members of the consensus. The **required** field is a comma-separated list of true/false values associated with each consensus member. If a member is designated as true, the member is required to be present in order for the consensus to be generated. If a member is false, the consensus will be generated regardless of whether the member is present. The length of the required array must be the same length as the members array. The **min_req** field is the number of members required in order for the consensus to be computed. The required and min_req field options are applied at each forecast lead time. If any member of the consensus has a non-valid position or intensity value, the consensus for that valid time will not be generated.

```
lag_time = [ “06”, “12” ];
```

The **lag_time** field is a comma-separated list of forecast lag times to be used in HH[MMSS] format. For each adeck track identified, a lagged track will be derived for each entry. In the tc_pairs output, the original adeck record will be retained, with the lagged entry listed as the adeck name with "_LAG_HH" appended.

```
best_technique = [ "BEST" ];
best_baseline  = [ "BCLP", "BCD5", "BCLA" ];
```

The **best_technique** field specifies a comma-separated list of technique name(s) to be interpreted as BEST track data. The default value (BEST) should suffice for most users. The **best_baseline** field specifies a comma-separated list of CLIPER/SHIFOR baseline forecasts to be derived from the best tracks. Specifying multiple **best_technique** values and at least one **best_baseline** value results in a warning since the derived baseline forecast technique names may be used multiple times.

The following are valid baselines for the **best_baseline** field:

BTCLIP: Neumann original 3-day CLIPER in best track mode. Used for the Atlantic basin only. Specify model as BCLP.

BTCLIP5: 5-day CLIPER (**Aberson, 1998**)/SHIFOR (**DeMaria and Knaff, 2001**) in best track mode for either Atlantic or eastern North Pacific basins. Specify model as BCS5.

BTCLIPA: Sim Aberson's recreation of Neumann original 3-day CLIPER in best-track mode. Used for Atlantic basin only. Specify model as BCLA.

```
oper_technique = [ "CARQ" ];
oper_baseline  = [ "OCLP", "OCS5", "OCD5" ];
```

The **oper_technique** field specifies a comma-separated list of technique name(s) to be interpreted as operational track data. The default value (CARQ) should suffice for most users. The **oper_baseline** field specifies a comma-separated list of CLIPER/SHIFOR baseline forecasts to be derived from the operational tracks. Specifying multiple **oper_technique** values and at least one **oper_baseline** value results in a warning since the derived baseline forecast technique names may be used multiple times.

The following are valid baselines for the **oper_baseline** field:

OCLIP: Merrill modified (operational) 3-day CLIPER run in operational mode. Used for Atlantic basin only. Specify model as OCLP.

OCLIP5: 5-day CLIPER (**Aberson, 1998**)/ SHIFOR (**DeMaria and Knaff, 2001**) in operational mode, rerun using CARQ data. Specify model as OCS5.

OCLIPD5: 5-day CLIPER (**Aberson, 1998**)/ DECAY-SHIFOR (**DeMaria and Knaff, 2001**). Specify model as OCD5.

```
anly_track = BDECK;
```

Analysis tracks consist of multiple track points with a lead time of zero for the same storm. An analysis track may be generated by running model analysis fields through a tracking algorithm. The **anly_track** field specifies which datasets should be searched for analysis track data and may be set to **NONE**, **ADECK**, **BDECK**, or **BOTH**. Use **BOTH** to create pairs using two different analysis tracks.

```
match_points = TRUE;
```

The **match_points** field specifies whether only those track points common to both the adeck and bdeck tracks should be written out. If **match_points** is selected as FALSE, the union of the adeck and bdeck tracks will be written out, with "NA" listed for unmatched data.

```
dland_file = "MET_BASE/tc_data/dland_global_tenth_degree.nc";
```

The **dland_file** string specifies the path of the NetCDF format file (default file: dland_global_tenth_degree.nc) to be used for the distance to land check in the **tc_pairs code**. This file is generated using tc_dland (default file provided in installed **share/met/tc_data** directory).

```
watch_warn = {
   file_name   = "MET_BASE/tc_data/wwpts_us.txt";
   time_offset = -14400;
}
```

The **watch_warn** field specifies the file name and time applied offset to the **watch_warn** flag. The **file_name** string specifies the path of the watch/warning file to be used to determine when a watch or warning is in affect during the forecast initialization and verification times. The default file is named **wwpts_us.txt**, which is found in the installed **share/met/tc_data/** directory within the MET build. The **time_offset** string is the time window (in seconds) assigned to the watch/warning. Due to the non-uniform time watches and warnings are issued, a time window is assigned for which watch/warnings are included in the verification for each valid time. The default watch/warn file is static, and therefore may not include warned storms beyond the current MET code release date; therefore users may wish to contact `met_help@ucar.edu` to obtain the most recent watch/warning file if the static file does not contain storms of interest.

20.2.3 tc_pairs output

TC-Pairs produces output in TCST format. The default output file name can be overwritten using the -out file argument in the usage statement. The TCST file output from TC-Pairs may be used as input into the TC-Stat tool. The header column in the TC-Pairs output is described in Table 20.1.

Table 20.1: Header information for TC-Pairs TCST output.

HEADER		
Column Number	**Header Column Name**	**Description**
1	VERSION	Version number
2	AMODEL	User provided text string designating model name
3	BMODEL	User provided text string designating model name
4	STORM_ID	BBCCYYY designation of storm
5	BASIN	Basin (BB in STORM_ID)
6	CYCLONE	Cyclone number (CC in STORM_ID)
7	STORM_NAME	Name of Storm
8	INIT	Initialization time of forecast in YYYYMMDD_HHMMSS format.
9	LEAD	Forecast lead time in HHMMSS format.
10	VALID	Forecast valid time in YYYYMMDD_HHMMSS format.
11	INIT_MASK	Initialization time masking grid applied
12	VALID_MASK	Valid time masking grid applied
13	LINE_TYPE	Output line type (TCMPR or PROBRI)

Table 20.2: Format information for TCMPR (Tropical Cyclone Matched Pairs) output line type.

TCMPR OUTPUT FORMAT		
Column Number	**Header Column Name**	**Description**
13	TCMPR	Tropical Cyclone Matched Pair line type
14	TOTAL	Total number of pairs in track
15	INDEX	Index of the current track pair
16	LEVEL	Level of storm classification
17	WATCH_WARN	HU or TS watch or warning in effect
18	INITIALS	Forecaster initials
19	ALAT	Latitude position of adeck model
20	ALON	Longitude position of adeck model
21	BLAT	Latitude position of bdeck model
22	BLON	Longitude position of bdeck model
23	TK_ERR	Track error of adeck relative to bdeck (nm)
24	X_ERR	X component position error (nm)
25	Y_ERR	Y component position error (nm)
26	ALTK_ERR	Along track error (nm)
27	CRTK_ERR	Cross track error (nm)
28	ADLAND	adeck distance to land (nm)
29	BDLAND	bdeck distance to land (nm)
30	AMSLP	adeck mean sea level pressure
31	BMSLP	bdeck mean sea level pressure
32	AMAX_WIND	adeck maximum wind speed
33	BMAX_WIND	bdeck maximum wind speed
34, 35	A/BAL_WIND_34	a/bdeck 34-knot radius winds in full circle
36, 37	A/BNE_WIND_34	a/bdeck 34-knot radius winds in NE quadrant
38, 39	A/BSE_WIND_34	a/bdeck 34-knot radius winds in SE quadrant
40, 41	A/BSW_WIND_34	a/bdeck 34-knot radius winds in SW quadrant
42, 43	A/BNW_WIND_34	a/bdeck 34-knot radius winds in NW quadrant
44, 45	A/BAL_WIND_50	a/bdeck 50-knot radius winds in full circle
46, 47	A/BNE_WIND_50	a/bdeck 50-knot radius winds in NE quadrant
48, 49	A/BSE_WIND_50	a/bdeck 50-knot radius winds in SE quadrant
50, 51	A/BSW_WIND_50	a/bdeck 50-knot radius winds in SW quadrant
52, 53	A/BNW_WIND_50	a/bdeck 50-knot radius winds in NW quadrant
54, 55	A/BAL_WIND_64	a/bdeck 64-knot radius winds in full circle
56, 57	A/BNE_WIND_64	a/bdeck 64-knot radius winds in NE quadrant
58, 59	A/BSE_WIND_64	a/bdeck 64-knot radius winds in SE quadrant
60, 61	A/BSW_WIND_64	a/bdeck 64-knot radius winds in SW quadrant
62, 63	A/BNW_WIND_64	a/bdeck 64-knot radius winds in NW quadrant
64, 65	A/BRADP	pressure in millibars of the last closed isobar, 900 - 1050 mb
66, 67	A/BRRP	radius of the last closed isobar in nm, 0 - 9999 nm
68, 69	A/BMRD	radius of max winds, 0 - 999 nm
70, 71	A/BGUSTS	gusts, 0 through 995 kts
72, 73	A/BEYE	eye diameter, 0 through 999 nm
74, 75	A/BDIR	storm direction in compass coordinates, 0 - 359 degrees
76, 77	A/BSPEED	storm speed, 0 - 999 kts
78, 79	A/BDEPTH	system depth, D-deep, M-medium, S-shallow, X-unknown

Table 20.3: Format information for PROBRIRW (Probability of Rapid Intensification) output line type.

PROBRIRW OUTPUT FORMAT		
Column Number	**Header Column Name**	**Description**
13	PROBRI	Probability of Rapid Intensification line type
14	ALAT	Latitude position of edeck model
15	ALON	Longitude position of edeck model
16	BLAT	Latitude position of bdeck model
17	BLON	Longitude position of bdeck model
18	INITIALS	Forecaster initials
19	TK_ERR	Track error of adeck relative to bdeck (nm)
20	X_ERR	X component position error (nm)
21	Y_ERR	Y component position error (nm)
22	ADLAND	adeck distance to land (nm)
23	BDLAND	bdeck distance to land (nm)
24	RI_BEG	Start of RI time window in HH format
25	RI_END	End of RI time window in HH format
26	RI_WINDOW	Width of RI time window in HH format
27	AWIND_END	Forecast maximum wind speed at RI end
28	BWIND_BEG	Best track maximum wind speed at RI begin
29	BWIND_END	Best track maximum wind speed at RI end
30	BDELTA	Exact Best track wind speed change in RI window
31	BDELTA_MAX	Maximum Best track wind speed change in RI window
32	BLEVEL_BEG	Best track storm classification at RI begin
33	BLEVEL_END	Best track storm classification at RI end
34	N_THRESH	Number of pro-ability thresholds
35	THRESH_i	The ith probability threshold value (repeated)
36	PROB_i	The ith probability value (repeated)

Chapter 21

TC-Stat Tool

21.1 Introduction

The TC-Stat tool ties together results from the TC-Pairs tool by providing summary statistics and filtering jobs on TCST output files. The TC-Stat tool requires TCST output from the TC-Pairs tool. See Section20.2.3 of this users guide for information on the TCST output format of the TC-Pairs tool. The TC-Stat tool supports several analysis job types. The **filter** job stratifies the TCST data using various conditions and thresholds described in Section21.3.2. The **summary** job produces summary statistics including frequency of superior performance, time-series independence calculations, and confidence intervals on the mean. The **rirw** job processes TCMPR lines, identifies adeck and bdeck rapid intensification or weakening events, populates a 2x2 contingency table, and derives contingency table statistics. The **probrirwjob** process PROBRIRW lines, populates an Nx2 probabilistic contingency table, and derives probabilistic statistics. The statistical aspects are described in Section21.2, and practical use information for the TC-Stat tool is described in Section21.3.

21.2 Statistical aspects

21.2.1 Filter TCST lines

The TC-Stat tool can be used to simply filter specific lines of the TCST file based on user-defined filtering criteria. All of the TCST lines that are retained from one or more files are written out to a single output file. The output file is also in TCST format.

Filtering options are outlined below in Section21.3.2 (configuration file). If multiple filtering options are listed, the job will be performed on their intersection.

21.2.2 Summary statistics for columns

The TC-Stat tool can be used to produce summary information for a single column of data. After the user specifies the specific column of interest, and any other relevant search criteria, summary information is produced from values in that column of data. The summary statistics produced are listed in Table 21.1.

Confidence intervals are computed for the mean of the column of data. Confidence intervals are computed using the assumption of normality for the mean. For further information on computing confidence intervals, refer to Appendix D of the MET user's guide.

When operating on columns, a specific column name can be listed (e.g. TK_ERR), as well as the differences of two columns (e.g. AMAX_WIND-BMAX_WIND), and the absolute difference of the column(s) (e.g. abs(AMAX_WIND-BMAX_WIND)). Additionally, several shortcuts can be applied to choose multiple columns with a single entry. Shortcut options for the -column entry are as follows:

TRACK: track error (TK_ERR), along-track error (ALTK_ERR), and cross-track error (CRTK_ERR)

WIND: all wind radii errors (34-, 50-, and 64-kt) for each quadrant

TI: track error (TK_ERR) and absolute intensity error (abs(AMAX_WIND-BMAX_WIND))

AC: along- and cross-track errors (ALTK_ERR, CRTK_ERR)

XY: X- and Y-component track errors (X_ERR, Y_ERR)

The TC-Stat tool can also be used to generate frequency of superior performance and the time to independence calculations when using the TC-Stat summary job.

Frequency of Superior Performance

The frequency of superior performance (FSP) looks at multiple model forecasts (adecks), and ranks each model relative to other model performance for the column specified in the summary job. The summary job output lists the total number of cases included in the FSP, the number of cases where the model of interest is the best (e.g.: lowest track error), the number of ties between the models, and the FSP (percent). Ties are not included in the FSP percentage; therefore the percentage may not equal 100%.

Time-Series Independence

The time-series independence evaluates effective forecast separation time using the Siegel method, by comparing the number of runs above and below the mean error to an expected value. This calculation expects the columns in the summary job to be a time series. The output includes the forecast hour interval and the number of hours to independence.

21.2.3 Rapid Intensification/Weakening

The TC-Stat tool can be used to read TCMPR lines and compare the occurrence of rapid intensification (i.e. increase in intensity) or weakening (i.e. decrease in intensity) between the adeck and bdeck. The rapid intensification or weakening is defined by the change of maximum wind speed (i.e. **AMAX_WIND** and **BMAX_WIND** columns) over a specified amount of time. Accurately forecasting large changes in intensity is a challenging problem and this job helps quantify a model's ability to do so.

Users may specify several job command options to configure the behavior of this job. Using these configurable options, the TC-Stat tool analyzes paired tracks and for each track point (i.e. each TCMPR line) determines whether rapid intensification or weakening occurred. For each point in time, it uses the forecast and BEST track event occurrence to populate a 2x2 contingency table. The job may be configured to require that forecast and BEST track events occur at exactly the same time to be considered a hit. Alternatively, the job may be configured to define a hit as long as the forecast and BEST track events occurred within a configurable time window. Using this relaxed matching criteria false alarms may be considered hits and misses may be considered correct negatives as long as the adeck and bdeck events were close enough in time. Each rirw job applies a single intensity change threshold. Therefore, assessing a model's performance with rapid intensification and weakening requires that two separate jobs be run.

21.2.4 Probability of Rapid Intensification

The TC-Stat tool can be used to accumulate multiple PROBRIRW lines and derive probabilistic statistics summarizing performance. The PROBRIRW line contains a probabilistic forecast for a specified intensity change along with the actual intensity change that occurred in the BEST track. Accurately forecast the likelihood of large changes in intensity is a challenging problem and this job helps quantify a model's ability to do so.

Users may specify several job command options to configure the behavior of this job. The TC-Stat tools reads the input PROBI lines, applies the configurable options to extract a forecast probability value and BEST track event, and bins those probabilistic pairs into an Nx2 contingency table. This job writes up to four probabilistic output line types summarizing the performance.

21.3 Practical information

The following sections describe the usage statement, required arguments, and optional arguments for **tc_stat**.

21.3.1 tc_stat usage

The usage statement for **tc_stat** is shown below:

```
Usage: tc_stat
       -lookin source
       [-out file]
       [-log file]
       [-v level]
       [-config file] | [JOB COMMAND LINE]
```

TC-Stat has one required argument and accepts optional ones.

The usage statement for the TC-Stat tool includes the "job" term, which refers to the set of tasks to be performed after applying user-specified filtering options. The filtering options are used to pare down the TC-Pairs output to only those lines that are desired for the analysis. The job and its filters together comprise a "job command line". The "job command line" may be specified either on the command line to run a single analysis job or within the configuration file to run multiple analysis jobs at the same time. If jobs are specified in both the configuration file and the command line, only the jobs indicated in the configuration file will be run. The various jobs are described in Table 21.1 and the filtering options are described in Section21.3.2.

Required arguments for tc_stat

1. The **-lookin source** argument indicates the location of the input TCST files generated from **tc_pairs**. This argument can be used one or more times to specify the name of a TCST file or top-level directory containing TCST files to be processed. Multiple tcst files may be specified by using a wild card (*).

2. Either a configuration file must be specified with the **-config** option, or a **JOB COMMAND LINE** must be denoted. The **JOB COMMAND LINE** options are described in Section 21.3.2,

Optional arguments for tc_stat

2. The **-out file** argument indicates the desired name of the TCST format output file.

3. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

4. The **-v level** option indicates the desired level of verbosity. The contents of “level” will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

5. The **-config file** argument indicates the name of the configuration file to be used. The contents of the configuration file are discussed below.

An example of the **tc_stat** calling sequence is shown below:

```
tc_stat -lookin /home/tc_pairs/*al092010.tcst -config TCStatConfig
```

In this example, the TC-Stat tool uses any TCST file (output from **tc_pairs**) in the listed directory for the 9th Atlantic Basin storm in 2010. Filtering options and aggregated statistics are generated following configuration options specified in the **TCStatConfig** file. Further, using flags (e.g. **-basin**, **-column, -storm_name, etc...)** option within the job command lines may further refine these selections. See Section 21.3.2 for options available for job command line and 3.5.2 for how to use them.

21.3.2 tc_stat configuration file

The default configuration file for the **TC-Stat** tool named **TCStatConfig_default** can be found in the installed share/met/config directory. Like the other configuration files described in this document, it is recommended that users make a copy of these files prior to modifying their contents.

The contents of the tc_stat configuration file are described below.

```
storm_id     = [];
basin        = [];
cyclone      = [];
storm_name   = [];
init_beg     = "";
init_end     = "";
init_inc     = [];
init_exc     = [];
valid_beg    = "";
valid_end    = "";
init_hour    = [];
lead_req     = [];
init_mask    = [];
valid_mask   = [];
match_points = TRUE;
version      = "VN.N";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.2.

Note that the options specified in the first section of the configuration file, prior to the job list, will be applied to every job specified in the joblist. However, if an individual job specifies an option listed above, it will be applied to that job. For example, if model = [**"GFSI", "LGEM", "DSHP"**]; is set at the top, but the job in the joblist sets the -model option to **"LGEM"**, that job will only run using the LGEM model data.

```
amodel = [];
bmodel = [];
```

The **amodel** and **bmodel** fields stratify by the amodel and bmodel columns based on a comma-separated list of model names used for all analysis performed. The names must be in double quotation marks (e.g.: "HWFI"). The **amodel** list specifies the model to be verified against the listed bmodel. The **bmodel** specifies the reference dataset, generally the BEST track analysis. Using the **-amodel** and **-bmodel** options within the job command lines may further refine these selections.

```
valid_inc = [];
valid_exc = [];
```

The **valid_inc** and **valid_exc** fields stratify by valid times, based on a comma-separated list of specific valid times to include (inc) or exclude (exc). Time strings are defined by YYYYMMDD[_HH[MMSS]]. Using the **-valid_inc** and **-valid_exc** options within the job command lines may further refine these selections.

```
valid_hour = [];
lead       = [];
```

The **valid_hour**, and **lead** fields stratify by the initialization time, valid time, and lead time, respectively. This field specifies a comma-separated list of initialization times, valid times, and lead times in HH[MMSS] format. Using the **-valid_hour** and **-lead** options within the job command lines may further refine these selections.

```
line_type = [];
```

The **line_type** field stratifies by the line_type column. Currently TCMPR is the only line_type option used in MET-TC.

```
track_watch_warn = [];
```

The **track_watch_warn** flag stratifies over the watch_warn column in the TCST files. If any of the watch/warning statuses are present in a forecast track, the entire track is verified. The value "ALL" matches HUWARN, HUWATCH, TSWARN, TSWATCH. Using the **-track_watch_warn** option within the job command lines may further refine these selections.

Other uses of the WATCH_WARN column include filtering when:

1. A forecast is issued when a watch/warn is in effect
2. A forecast is verifying when a watch/warn is in effect
3. A forecast is issued when a watch/warn is NOT in effect
4. A forecast is verified when a watch/warn is NOT in effect

The following filtering options can be achieved by the following:

```
1. init_str_name = ["WATCH_WARN"];
    init_str_val = ["ALL"];
2. column_str_name = ["WATCH_WARN"];
    column_str_val = ["ALL"];
3. init_str_name = ["WATCH_WARN"];
    init_str_val = ["NA"];
4. column_str_name = ["WATCH_WARN"];
    column_str_val = ["NA"];
```

Further information on the **column_str** and **init_str** fields is described below. Listing a comma-separated list of watch/warning types in the **column_str_val** field will stratify by a single or multiple types of warnings.

```
column_thresh_name = [];
column_thresh_val  = [];
```

The **column_thresh_name** and **column_thresh_val** fields stratify by applying thresholds to numeric data columns. Specify a comma-separated list of column names and thresholds to be applied. The length of **column_thresh_val** should match that of **column_thresh_name**. Using the **-column_thresh name thresh** option within the job command lines may further refine these selections.

```
column_str_name = [];
column_str_val  = [];
```

The **column_str_name** and **column_str_val** fields stratify by performing string matching on non-numeric data columns. Specify a comma-separated list of columns names and values to be checked. The length of the **column_str_val** should match that of the **column_str_name**. Using the **-column_str name val** option within the job command lines may further refine these selections.

```
init_thresh_name = [];
init_thresh_val  = [];
```

The **init_thresh_name** and **init_thresh_val** fields stratify by applying thresholds to numeric data columns only when lead = 0. If lead =0, but the value does not meet the threshold, discard the entire track. The length of the **init_thresh_val** should match that of the **init_thresh_name**. Using the **-init_thresh name val** option within the job command lines may further refine these selections.

```
init_str_name = [];
init_str_val  = [];
```

The **init_str_name** and **init_str_val** fields stratify by performing string matching on non-numeric data columns only when lead = 0. If lead =0, but the string does not match, discard the entire track. The length of the **init_str_val** should match that of the **init_str_name**. Using the **-init_str name val** option within the job command lines may further refine these selections.

```
water_only = FALSE;
```

The **water_only** flag stratifies by only using points where both the amodel and bmodel tracks are over water. When **water_only = 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.

```
rirw = {
   track  = NONE;
   time   = "24";
   exact  = TRUE;
   thresh = >=30.0;
}
```

The **rirw** field specifies those track points for which rapid intensification (RI) or rapid weakening (RW) occurred, based on user defined RI/RW thresholds. The **track** entry specifies that RI/RW is not turned on (**NONE**), is computed based on the bmodel only (**BDECK**), is computed based on the amodel only (**ADECK**), or computed when both the amodel and bmodel (the union of the two) indicate RI/RW (BOTH). If **track** is set to **ADECK, BDECK,** or **BOTH**, only tracks exhibiting rapid intensification will be retained. Rapid intensification is officially defined as when the change in the maximum wind speed over a 24-hour period is greater than or equal to 30 kts. This is the default setting, however flexibility in this definition is provided through the use of the **time, exact** and **thresh** options. The **time** field specifies the time window (HH[MMSS] format) for which the RI/RW occurred. The **exact** field specifies whether to only count RI/RW when the intensity change is over the exact time window (TRUE), which follows the official RI definition, or if the intensity threshold is met anytime during the time window (FALSE). Finally, the **thresh** field specifies the user defined intensity threshold (where ">=" indicates RI, and "<=" indicates RW).

Using the **-rirw_track**, **-rirw_time_adeck**, **-rirw_time_bdeck**, **-rirw_exact_adeck**, **-rirw_exact_bdeck**, **-rirw_thresh_adeck**, **-rirw_thresh_bdeck** options within the job command lines may further refine these selections. See README_TC in data/config for how to use these options.

```
landfall     = FALSE;
landfall_beg = "-24";
landfall_end = "00";
```

The **landfall, landfall_beg,** and **landfall_end** fields specify whether only those track points occurring near landfall should be retained. The landfall retention window is defined as the hours offset from the time of landfall. Landfall is defined as the last bmodel track point before the distance to land switches from water to land. When **landfall_end** is set to 0, the track is retained from the **landfall_beg** to the time of landfall. Using the **-landfall_window** option with the job command lines may further refine these selections. The **-landfall_window** job command option takes 1 or 2 arguments in HH[MMSS] format. Use 1 argument to define a symmetric time window. For example, **-landfall_window 06** defines the time window +/- 6 hours around the landfall time. Use 2 arguments to define an asymmetric time window. For example, **-landfall_window 00 12** defines the time window from the landfall event to 12 hours after.

```
event_equal = FALSE;
```

The **event_equal** flag specifies whether only those track points common to all models in the dataset should be retained. The event equalization is performed only using cases common to all listed amodel entries. A case is defined by comparing the following columns in the TCST files: BMODEL, BASIN, CYCLONE, INIT, LEAD, VALID. This option may be modified using the **-event_equal** option within the job command lines.

```
event_equal_lead = [];
```

The **event_equal_lead** flag specifies lead times that must be present for a track to be included in the event equalization logic. The event equalization is performed only using cases common to all lead times listed, enabling the verification at each lead time to be performed on a consistent dataset. This option may be modified using the **-event_equal_lead** option within the job command lines.

```
out_init_mask = "";
```

The **out_init_mask** field applies polyline masking logic to the location of the amodel track at the initialization time. If the track point falls outside the mask, discard the entire track. This option may be modified using the **-out_init_mask** option within the job command lines.

```
out_valid_mask = "";
```

The **out_valid_mask** field applies polyline masking logic to the location of the amodel track at the valid time. If the track point falls outside the mask, discard the entire track. This option may be modified using the **-out_valid_mask** option within the job command lines.

```
jobs = [];
```

The user may specify one or more analysis jobs to be performed on the TCST lines that remain after applying the filtering parameters listed above. Each entry in the joblist contains the task and additional filtering options for a single analysis to be performed. There are three types of jobs available including *filter, summary, and rirw.* Please refer to the README_TC in data/config for details on how to call each job. The format for an analysis job is as follows:

```
-job job_name REQUIRED and OPTIONAL ARGUMENTS
e.g.: -job filter  -line_type TCMPR  -amodel HWFI   -dump_row ./tc_filter_job.tcst
      -job summary -line_type TCMPR  -column TK_ERR -dump_row ./tc_summary_job.tcst
      -job rirw    -line_type TCMPR  -rirw_time 24 -rirw_exact false -rirw_thresh ge20
      -job probrirw -line_type PROBRIRW -column_thresh RI_WINDOW ==24 \
                    -probri_thresh 30 -probri_prob_thresh ==0.25
```

21.3.3 tc_stat output

The output generated from the TC-Stat tool contains statistics produced by the analysis. Additionally, it includes information about the analysis job that produced the output for each line. The output can be redirected to an output file using the **-out** option. The format of output from each **tc_stat** job command is listed below.

Job: Filter

This job command finds and filters TCST lines down to those meeting the criteria selected by the filter's options. The filtered TCST lines are written to a file specified by the **-dump_row** option. The TCST output from this job follows the TCST output description in Chapters 19 and 20.

Job: Summary

This job produces summary statistics for the column name specified by the **-column** option. The output of the summary job consists of three rows: "JOB_LIST", which shows the job definition parameters used for this job. "COL_NAME", followed by the summary statistics that are applied. "SUMMARY", which is followed by the total, mean (with confidence intervals), standard deviation, minimum value, percentiles (10th, 25th, 50th, 75th, 90th), maximum value, interquartile range, range, sum, time to independence, and frequency of superior performance. The output columns are shown below in Table 21.1 The **-by** option can also be used one or more times to make this job more powerful. Rather than running the specified job once, it will be run once for each unique combination of the entries found in the column(s) specified with the **-by** option.

Table 21.1: Columnar output of "summary" job output from the TC-Stat tool.

tc_stat Summary Job Output Options	
Column number	**Description**
1	SUMMARY: (job type)
2	Column (dependent parameter)
3	Case (storm + valid time)
4	Total
5	Valid
6-8	Mean including normal upper and lower confidence limits
9	Standard deviation
10	Minimum value
11-15	Percentiles (10th, 25th, 50th, 75th, 90th)
16	Maximum Value
17	Interquartile range (75th - 25th percentile)
18	Range (Maximum - Minimum)
19	Sum
20-21	Independence time
22-25	Frequency of superior performance

Job: RIRW

The RIRW job produces contingency table counts and statistics defined by identifying rapid intensification or weakening events in the adeck and bdeck track. Users may specify several job command options to configure the behavior of this job:

- The **-rirw_time HH[MMSS]** option (or **-rirw_time_adeck** and **-rirw_time_bdeck** to specify different settings) defines the time window of interest. The default is 24 hours.
- The **-rirw_exact bool** option (or **-rirw_exact_adeck** and **-rirw_exact_bdeck** to specify different settings) is a boolean defining whether the exact intensity change or maximum intensity change over that time window should be used. For rapid intensification, the maximum increase in computed. For rapid weakening, the maximum decrease is used. The default is true.
- The **-rirw_thresh threshold** option (or **-rirw_thresh_adeck** and **-rirw_thresh_bdeck** to specify different settings) defines the intensity change event threshold. The default is greater than or equal to 30 kts.
- The **-rirw_window** option may be passed one or two arguments in **HH[MMSS]** format to define how close adeck and bdeck events must be to be considered hits or correct negatives. One time string defines a symmetric time window while two time strings define an asymmetric time window. The default is 0, requiring an exact match in time.
- The **-out_line_type** option defines the output data that should written. This job can write contingency table counts (CTC), contingency table statistics (CTS), and RIRW matched pairs (MPR). The default is CTC and CTS, but the MPR output provides great amount of detail.

Users may also specify the **-out_alpha** option to define the alpha value for the confidence intervals in the CTS output line type. In addition, the **-by column_name** option is a convenient way of running the same job across multiple stratifications of data. For example, **-by AMODEL** runs the same job for each unique AMODEL name in the data.

Job: PROBRIRW

The PROBRIRW job produces probabilistic contingency table counts and statistics defined by placing forecast probabilities and BEST track rapid intensification events into an Nx2 contingency table. Users may specify several job command options to configure the behavior of this job:

- The **-prob_thresh n** option is required and defines which probability threshold should be evaluated. It determines which **PROB_i** column from the PROBRIRW line type is selected for the job. For example, use **-prob_thresh 30** to evaluate forecast probabilities of a 30 kt increase or use **-prob_thresh -30** to evaluate forecast probabilities of a 30 kt decrease in intensity. The default is a 30 kt increase.
- The **-prob_exact bool** option is a boolean defining whether the exact or maximum BEST track intensity change over the time window should be used. If true, the values in the **BDELTA** column are used. If false, the values in the **BDELTA_MAX** column are used. The default is true.
- The **-probri_bdelta_thresh threshold** option defines the BEST track intensity change event threshold. This should typically be set consistent with the probability threshold (**-prob_thresh**) chosen above. The default is greater than or equal to 30 kts.

- The **-probri_prob_thresh threshold_list** option defines the probability thresholds used to create the output Nx2 contingency table. The default is probability bins of width 0.1. These probabilities may be specified as a list (>0.00,>0.25,>0.50,>0.75,>1.00) or using shorthand notation (==0.25) for bins of equal width.

- The **-out_line_type** option defines the output data that should written. This job can write PCT, PSTD, PJC, and PRC output line types. The default is PCT and PSTD.

Users may also specify the **-out_alpha** option to define the alpha value for the confidence intervals in the PSTD output line type. Multiple values in the **RI_WINDOW** column cannot be combined in a single PROBRIRW job since BEST track intensity threshold should change for each. Using the **-by RI_WINDOW** option or **-column_thresh RI_WINDOW ==24** option provide convenient ways avoiding this problem.

Users should note that for the PROBRIRW line type, **PROBRI_PROB** is a derived column name. The -probri_thresh option defines the probabilities of interest (e.g. **-probri_thresh 30**) and the **PROBRI_PROB** column name refers those probability values, regardless of their column number. For example, the job command options **-probri_thresh 30 -column_thresh PROBRI_PROB >0** select 30 kt probabilities and match probability values greater than 0.

Chapter 22

TC-Gen Tool

22.1 Introduction

The TC-Gen tool provides verification of tropical cyclone genesis forecasts in ATCF file format. Producing reliable tropical cyclone genesis forecasts is an important metric for global numerical weather prediction models. This tool ingests deterministic model output post-processed by a genesis tracking software (e.g. GFDL vortex tracker) and ATCF format reference dataset(s) (e.g. Best Track analysis and CARQ operational tracks) and outputs categorical counts and statistics. The capability to modify the spatial and temporal tolerances that define a "hit" forecast is included to give users the ability to condition the criteria based on model performance and/or conduct sensitivity analyses. Statistical aspects are outlines in Section 21.2 and practical aspects of the TC-Gen tool are described in Section 21.3.

22.2 Statistical aspects

The TC-Gen tool populates a contingency table with hits, misses, and false alarms. As with other extreme events (where the event occurs much less frequently than the non-event), the correct negative category is not computed the non-events would dominate the contingency table. Therefore, only statistics that do not include correct negatives should be considered for this tool. The following CTS statistics are relevant: Base rate (BASER), Mean forecast (FMEAN), Frequency Bias (FBIAS), Probability of Detection (PODY), False Alarm Ratio (FAR), Critical Success Index (CSI), Gilbert Skill Score (GSS), Extreme Dependency Score (EDS), Symmetric Extreme Dependency Score (SEDS), Bias Adjusted Gilbert Skill Score (BAGSS).

Other considerations for interpreting the output of the TC-Gen tool involve the size of the contingency table output. The size of the contingency table will change depending on the number of matches. Additionally, the number of misses is based on the forecast duration and interval (specified in the configuration file). This change is due to the number of model opportunities to forecast the event, which is determined by the specified duration/interval.

Care should be taken when interpreting the statistics for filtered data. In some cases, variables (e.g. storm name) are only available in either the forecast or reference datasets, rather than both. When filtering on a field that is only present in one dataset, the contingency table counts will be impacted. Similarly, the initialization field only impacts the model forecast data. If the valid time (which will impact the reference dataset) isn't also specified, the forecasts will be filtered and matched such that the number of misses will erroneously increase. See section 1.3.2 for more detail.

22.3 Practical information

This section describes how to configure and run the TC-Gen tool. The TC-Gen tool identifies tropical cyclone genesis events in both genesis forecasts and ATCF track datasets. It applies configurable logic to process the forecast and observed genesis events, classify them, and populate a contingency table with hits, misses, and false alarms. It writes the categorical counts and statistics to the output file(s). The tool can be configured to apply one or more sets of filtering criteria in a single run. The following sections describe the usage statement, required arguments, and optional arguments for **tc_gen**.

22.3.1 tc_gen usage

The usage statement for **tc_gen** is shown below:

```
Usage: tc_gen
       -genesis path
       -track path
       -config file
       [-out base]
       [-log file]
       [-v level]
```

TC-Gen has three required arguments and accepts optional ones.

Required arguments for tc_gen

1. The **-genesis path** argument specifies an ATCF forecast genesis file or top-level directory containing them, with files matching the regular expression "atcf_gen". The **-genesis** option must be used at least once.

2. The **-track path** argument specifies an ATCF track file or top-level directory containing them, with files ending in ".dat". This tool processes both BEST track data from bdeck files and operational track data (e.g. CARQ) from adeck files. Both adeck and bdeck data should be provided using the **-track** option. The **-track** option must be used at least once.

3. The **-config file** argument indicates the name of the configuration file to be used. The contents of the configuration file are discussed below.

Optional arguments for tc_gen

4. The **-out base** argument indicates the path of the output file base. This argument overrides the default output file base (./tc_gen)

5. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no log file.

6. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

The TC-Gen tool implements the following logic:

- Parse the genesis data and identify forecast genesis events separately for each model present.
- Parse the BEST and operational track data and identify observed genesis events.
- Loop over the filters defined in the configuration file and apply the following logic for each.
  - For each forecast genesis event, search the BEST genesis events for a match that is close enough in time and space. If not found, search the operational genesis events for a match. If a match is found, classify the forecast genesis event as a **hit**. Otherwise, classify it as a **false alarm**.
  - For each BEST track genesis event, determine the initialization and lead times for which the model had an opportunity to forecast that genesis event. Classify as a **miss** each model opportunity that was not counted as a hit in the previous step.
  - Do not count any correct negatives.
- Report the contingency table hits, misses, and false alarms separately for each forecast model and configuration file filter.

22.3.2 tc_gen configuration file

The default configuration file for the **TC-Gen** tool named **TCGenConfig_default** can be found in the installed share/met/config directory. Like the other configuration files described in this document, it is recommended that users make a copy of these files prior to modifying their contents.

The tc_gen configuration file is divided into three main sections: criteria to define genesis events, options to subset and filter those events, and options to control the output. The contents of this configuration file are described below.

```
init_freq = 6;
```

The **init_freq** variable is an integer specifying the model initialization frequency in hours, starting at 00Z. The default value of 6 indicates that the model is initialized every day at 00Z, 06Z, 12Z, and 18Z. The same frequency is applied to all models processed. Models initialized at different frequencies should be processed with separate calls to tc_gen. The initialization frequency is used when defining the model opportunities to forecast the BEST track genesis events.

```
lead_window = {
   beg = 24;
   end = 120;
}
```

The **lead_window** option is a dictionary defining the beginning (**beg**) and ending (**end**) model forecast hours to be searched for genesis events. Model genesis events occurring outside of this window are ignored. This lead window is also used when defining the model opportunities to forecast the BEST track genesis events.

```
min_duration = 12;
```

The **min_duration** variable is an integer specifying the minimum number of hours a track must persist for its initial point to be counted as a genesis event. Some models spin up many short-lived storms, and this setting enables them to be excluded from the analysis.

```
fcst_genesis = {
   vmax_thresh = NA;
   mslp_thresh = NA;
}
```

The **fcst_genesis** dictionary defines the conditions required for a model track's genesis point to be included in the analysis. Thresholds for the maximum wind speed (**vmax_thresh**) and minimum sea level pressure (**mslp_thresh**) may be defined. These conditions must be satisfied for at least one track point for the genesis event to be included in the analysis. The default thresholds (**NA**) always evaluate to true.

```
best_genesis = {
   technique   = "BEST";
   category    = [ "TD", "TS" ];
   vmax_thresh = NA;
   mslp_thresh = NA;
}
oper_genesis = {
   technique   = "CARQ";
   category    = [ "DB", "LO", "WV" ];
   vmax_thresh = NA;
   mslp_thresh = NA;
}
```

The **best_genesis** and **oper_genesis** dictionaries defines genesis criteria for the BEST and operational tracks, respectively. Like the **fcst_genesis** dictionary, the **vmax_thresh** and **mslp_thresh** thresholds define required genesis criteria. In addition, the **category** array defines the ATCF storm categories that should qualify as genesis events. The **technique** string defines the ATCF ID for the BEST and operational tracks.

```
filter = [];
```

The **filter** entry is an array of dictionaries defining genesis filtering criteria to be applied. Each of the entries listed below (from **desc** to **genesis_radius**) may be specified separately within each filter dictionary. If left empty, the default setting, a single filter is applied using the top-level filtering criteria. If multiple filtering dictionaries are defined, the **desc** entry must be specified for each to differentiate the output data. Output is written for each combination of filter dictionary and model ATCF ID encountered in the data.

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

The **desc** configuration option is common to many MET tools and is described in Section 3.5.1.

```
model = [];
```

The **model** entry is an array defining the model ATCF ID's for which output should be computed. If left empty, the default setting, output will be computed for each model encountered in the data. Otherwise, output will be computed only for the ATCF ID's listed.

```
storm_id   = [];
storm_name = [];
```

The **storm_id** and **storm_name** entries are arrays indicating the ATCF storm ID's and storm names to be processed. If left empty, all tracks will be processed. Otherwise, only those tracks which meet these criteria will be included. Note that these strings only appear in the BEST and operational tracks, not the forecast genesis data. Therefore, these filters only apply to the BEST and operational tracks. Care should be given when interpreting the contingency table results for filtered data.

```
init_beg = "";
init_end = "";
```

The **init_beg** and **init_end** entries are strings in YYYYMMDD[_HH[MMSS]] format which defining which forecast and operational tracks initializations to be processed. If left empty, all tracks will be used. Otherwise, only those tracks whose initialization time falls within the window will be included. Note that these settings only apply to the forecast and operational tracks, not the BEST tracks, for which the initialization time is undefined. Care should be given when interpreting the contingency table results for filtered data.

```
valid_beg = "";
valid_end = "";
```

The **valid_beg** and **valid_end** entries are similar to **init_beg** and **init_end**, described above. However, they are applied to all genesis data sources. Only those tracks falling completely inside this window are included in the analysis.

```
init_hour = [];
lead      = [];
```

The **init_hour** and **lead** entries are arrays of strings in HH[MMSS] format defining which forecast and operational tracks should be included. If left empty, all tracks will be used. Otherwise, only those forecast and operational tracks whose initialization hour and lead times appear in the list will be used. Note that these settings only apply to the forecast and operational tracks, not the BEST tracks, for which the initialization time is undefined. Care should be given when interpreting the contingency table results for filtered data.

```
vx_mask = "MET_BASE/tc_data/basin_global_tenth_degree.nc \
           { 'name=\"basin\";level=\"(*,*)\"; } ==1";
```

The **vx_mask** entry is a string defining the path to a Lat/Lon polyline file or a gridded data file that MET can read to subset the results spatially. If specified, only those genesis events whose Lat/Lon location falls within the specified area will be included. The MET code includes the file **basin_global_tenth_degree.nc**, which contains a global definition of the Regional Specialized Meteorology Centers (RSMC) and hurricane basin regions. The above example uses this file to stratify genesis results for the Atlantic Basin, where the **basin** variable equals ones.

```
dland_thresh = NA;
```

The **dland_thresh** entry is a threshold defining whether the genesis event should be included based on it's distance to land. The default threshold (**NA**) always evaluate to true.

```
genesis_window = {
   beg = -24;
   end =  24;
}
```

The **genesis_window** entry defines a matching time window, in hours, relative to the forecast genesis time. When searching for a match, only those BEST/operational genesis events which occur within this time window will be considered. Increasing this time window should lead to an increase in hits.

```
genesis_radius = 300;
```

The **genesis_radius** entry defines a search radius, in km, relative to the forecast genesis location. When searching for a match, only those BEST/operational genesis events which occur within this radius will be considered. Increasing this search radius should lead to an increase in hits.

```
ci_alpha = 0.05;
output_flag = {
   fho = BOTH;
```

```
      ctc = BOTH;
      cts = BOTH;
   }
   dland_file = "MET_BASE/tc_data/dland_global_tenth_degree.nc";
   version    = "V9.0";
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1. Note that TC-Gen writes output for 2x2 contingency tables to the **FHO**, **CTC**, and **CTS** line types.

22.3.3 tc_gen output

TC-Gen produces output in STAT and, optionally, ASCII format. The ASCII output duplicates the STAT output but has the data organized by line type. The output files are created based on the **-out** command line argument. The default output base name, "./tc_gen" writes output files in the current working directory named "tc_gen.stat" and, optionally, "tc_gen_fho.txt", "tc_gen_ctc.txt", and "tc_gen_cts.txt". The contents of these output files are described in section 7.3.3.

Like all STAT output, the output of TC-Gen may be further processed using the Stat-Analysis tool, described in chapter 12.

Chapter 23

TC-RMW Tool

23.1 Introduction

The TC-RMW tool regrids tropical cyclone model data onto a moving range-azimuth grid centered on points along the storm track. The radial grid spacing may be set as a factor of the radius of maximum winds (RMW). If wind fields are specified in the configuration file the radial and tangential wind components will be computed. Any regridding method available in MET can be used to interpolate data on the model output grid to the specified range-azimuth grid. The regridding will be done separately on each vertical level. The model data files must coincide with track points in a user provided ATCF formatted track file.

23.2 Practical information

23.2.1 tc_rmw usage

The following sections describe the usage statement, required arguments, and optional arguments for **tc_rmw**.

```
Usage: tc_rmw
       -data file_1 ... file_n | data_file_list
       -adeck file
       -config file
       -out file
       [-log file]
       [-v level]
```

tc_rmw has required arguments and can accept several optional arguments.

Required arguments for tc_rmw

1. The **-data file_1 ... file_n | data_file_list** options specify the gridded data files or an ASCII file containing a list of files to be used.

2. The **-adeck source** argument is the adeck ATCF format data source.

3. The **-config file** argument is the configuration file to be used. The contents of the configuration file are discussed below.

4. The **-out** argument is the NetCDF output file to be written.

Optional arguments for tc_rmw

5. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no logfile.

6. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

23.2.2 tc_rmw configuration file

The default configuration file for the TC-RMW tool named 'TCRMWConfig_default' can be found in the installed **share/met/config/ directory**. It is encouraged for users to copy these default files before modifying their contents. The contents of the configuration file are described in the subsections below.

```
model         = "GFS";
censor_thresh = [];
censor_val    = [];
data  = {
   field = [
        {
           name = "PRMSL";
           level = ["L0"];
        },
        {
           name = "TMP";
           level = ["P1000", "P500"];
        },
        {
           name = "UGRD";
           level = ["P1000", "P500"];
        },
```

```
        {
           name = "VGRD";
           level = ["P1000", "P500"];
        }
    ];
}
regrid = { ... }
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
n_range = 100;
```

The **n_range** parameter is the number of equally spaced range intervals in the range-azimuth grid.

```
n_azimuth = 180;
```

The **n_azimuth** parameter is the number of equally spaced azimuth intervals in the range-azimuth grid. The azimuthal grid spacing is 360 / **n_azimuth** degrees.

```
max_range_km = 100.0;
```

The **max_range_km** parameter specifies the maximum range of the range-azimuth grid, in kilometers. If this parameter is specified and not **rmw_scale**, the radial grid spacing will be **max_range_km** / **n_range**.

```
delta_range_km = 10.0;
```

The **delta_range_km** parameter specifies the spacing of the range rings, in kilometers.

```
rmw_scale = 0.2;
```

The **rmw_scale** parameter overrides the **max_range_km** parameter. When this is set the radial grid spacing will be **rmw_scale** in units of the RMW, which varies along the storm track.

23.2.3 tc_rmw output file

The NetCDF output file contains the following dimensions:

1. *range* - the radial dimension of the range-azimuth grid
2. *azimuth* - the azimuthal dimension of the range-azimuth grid
3. *pressure* - if any pressure levels are specified in the data variable list, they will be sorted and combined into a 3D NetCDF variable, which *pressure* as the vertical dimension and *range* and *azimuth* as the horizontal dimensions
4. *track_point* - the track points corresponding to the model output valid times

For each data variable specified in the data variable list, a corresponding NetCDF variable will be created with the same name and units.

Chapter 24

RMW-Analysis Tool

24.1 Introduction

The RMW-Analysis tool analyzes a set of output files generated by the tc_rmw tool. For each grid cell it aggregates variable statistics across the set and across the track points of the tc_rmw output files. The statistics are mean, standard deviation, minimum and maximum. Note that tc_rmw should be set to use the same scale factor of the radius of maximum winds (RMW) as the unit of range for its range-azimuth grid. The specified data variables on the range-azimuth-vertical grid then share a common range scale of RMW before aggregation by rmw_analysis.

24.2 Practical information

24.2.1 rmw_analysis usage

The following sections describe the usage statement, required arguments, and optional arguments for **rmw_analysis**.

```
Usage: rmw_analysis
       -data file_1 ... file_n | data_file_list
       -config file
       -out file
       [-log file]
       [-v level]
```

rmw_analysis has required arguments and can accept several optional arguments.

Required arguments for rmw_analysis

1. The **-data file_1 ... file_n | data_file_list** argument is the NetCDF output of TC-RMW to be processed or an ASCII file containing a list of files.

2. The **-config file** argument is the RMWAnalysisConfig to be used. The contents of the configuration file are discussed below.

3. The **-out file** argument is the NetCDF output file to be written.

Optional arguments for rmw_analysis

4. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no logfile.

5. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

24.2.2 rmw_analysis configuration file

The default configuration file for the RMW-Analysis tool named 'RMWAnalysisConfig_default' can be found in the installed **share/met/config/ directory**. It is encouraged for users to copy these default files before modifying their contents. The contents of the configuration file are described in the subsections below.

```
model = "GFS";
data  = {
   level = [ "" ];
   field = [
      { name = "PRMSL"; },
      { name = "TMP";   }
   ];
}
```

The configuration options listed above are common to many MET tools and are described in Section 3.5.1.

```
basin      = "";
storm_name = "";
storm_id   = "";
cyclone    = "";
init_beg   = "";
```

```
init_end   = "";
valid_beg  = "";
valid_end  = "";
init_mask  = "";
valid_mask = "";
version    = "VN.N";
```

The track filter options available in rmw_analysis and listed above are described in Section 3.5.2.

24.2.3 rmw_analysis output file

The NetCDF output file will inherit the spatial grid from the first tc_rmw output file in the output file list. All tc_rmw files in this list must have the same grid dimension sizes. A NetCDF output error will result if that is not the case. For each data variable specified in the config file, four corresponding NetCDF variables will be written, e.g. TMP_mean, TMP_stdev, TMP_min, TMP_max. No track point dimension is retained in the rmw_analysis output.

Chapter 25

Plotting and Graphics Support

25.1 Plotting Utilities

This section describes how to check your data files using plotting utilities. Point observations can be plotted using the plot_point_obs utility. A single model level can be plotted using the plot_data_plane utility. For object based evaluations, the MODE objects can be plotted using plot_mode_field. Occasionally, a post-processing or timing error can lead to errors in MET. These tools can assist the user by showing the data to be verified to ensure that times and locations match up as expected.

25.1.1 plot_point_obs usage

The usage statement for the plot_point_obs utility is shown below:

```
Usage: plot_point_obs
       nc_file
       ps_file
       [-gc code]
       [-msg_typ name]
       [-data_file name]
       [-dotsize val]
       [-log file]
       [-v level]
```

plot_point_obs has two required arguments and can take optional ones.

Required arguments for plot_point_obs

1. The **nc_file** argument indicates the name of the file to be plotted.
2. The **ps_file** argument indicates the name given to the output file containing the plot.

Optional arguments for plot_point_obs

3. The **-gc code** is the GRIB code(s) to be plotted.
4. The **-msg_typ name** is the message type(s) to be plotted.
5. The **-data_file name** is a data file whose grid should be used for the plot.
6. The **-dotsize val** option overrides the default dot size value (1).
7. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no logfile.
8. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity will increase the amount of logging.

An example of the plot_point_obs calling sequence is shown below:

```
plot_point_obs sample_pb.nc sample_data.ps
```

In this example, the plot_point_obs tool will process the input sample_pb.nc file write a postscript file containing a plot to a file named sample_pb.ps.

25.1.2 plot_data_plane usage

The usage statement for the plot_data_plane utility is shown below:

```
Usage: plot_data_plane
       input_filename
       output_filename
       field_string
       [-color_table color_table_name]
       [-plot_range min max]
       [-title title_string]
       [-log file]
       [-v level]
```

plot_data_plane has two required arguments and can take optional ones.

Required arguments for plot_data_plane

1. The **input_filename** argument indicates the name of the gridded data file to be plotted.
2. The **output_filename** argument indicates the name given to the output PostScript file containing the plot.
3. The **field_string** argument contains information about the field and level to be plotted.

Optional arguments for plot_data_plane

4. The **-color_table color_table_name** overrides the default color table ("MET_BASE/colortables/met_default.ctable")
5. The **-plot_range min max** sets the minimum and maximum values to plot.
6. The **-title title_string** sets the title text for the plot.
7. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no logfile.
8. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity will increase the amount of logging.

An example of the plot_data_plane calling sequence is shown below:

```
plot_data_plane test.grb test.ps 'name="TMP"; level="Z2";'
```

A second example of the plot_data_plane calling sequence is shown below:

```
plot_data_plane test.grb2 test.ps 'name="DSWRF"; level="L0";' -v 4
```

In the first example, the plot_data_plane tool will process the input test.grb file and write a PostScript image to a file named test.ps showing temperature at 2 meters. The second example plots downward shortwave radiation flux at the surface. The second example is run at verbosity level 4 so that user can inspect the output and make sure its plotting the intended record.

25.1.3 plot_mode_field usage

The usage statement for the plot_mode_field utility is shown below:

```
Usage: plot_mode_field
       mode_nc_file_list
       -raw | -simple | -cluster
       -obs | -fcst
       -config file
       [-log file]
       [-v level]
```

plot_mode_field has four required arguments and can take optional ones.

Required arguments for plot_mode_field

1. The **mode_nc_file_list** specifies the MODE output files to be used for plotting.
2. The **-raw** | **-simple** | **-cluster** argument indicates the types of field to be plotted. Exactly one must be specified. For details about the types of objects, see the chapter in this document on MODE.
3. The **-obs** | **-fcst** option specifies whether to plot the observed or forecast field from the MODE output files. Exactly one must be specified.
4. The **-config file** specifies the configuration file to use for specification of plotting options.

Optional arguments for plot_mode_field

5. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as standard out and error. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no logfile.
6. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity will increase the amount of logging.

An example of the plot_mode_field calling sequence is shown below:

```
plot_mode_field -simple -obs -config \
plotMODEconfig mode_120000L_20050807_120000V_000000A_obj.nc
```

In this example, the plot_mode_field tool will plot simple objects from an observed precipitation field using parameters from the configuration file plotMODEconfig and objects from the MODE output file mode_120000L_20050807_120000V_000000A_obj.nc. An example plot showing twelve simple observed precipitation objects is shown below.

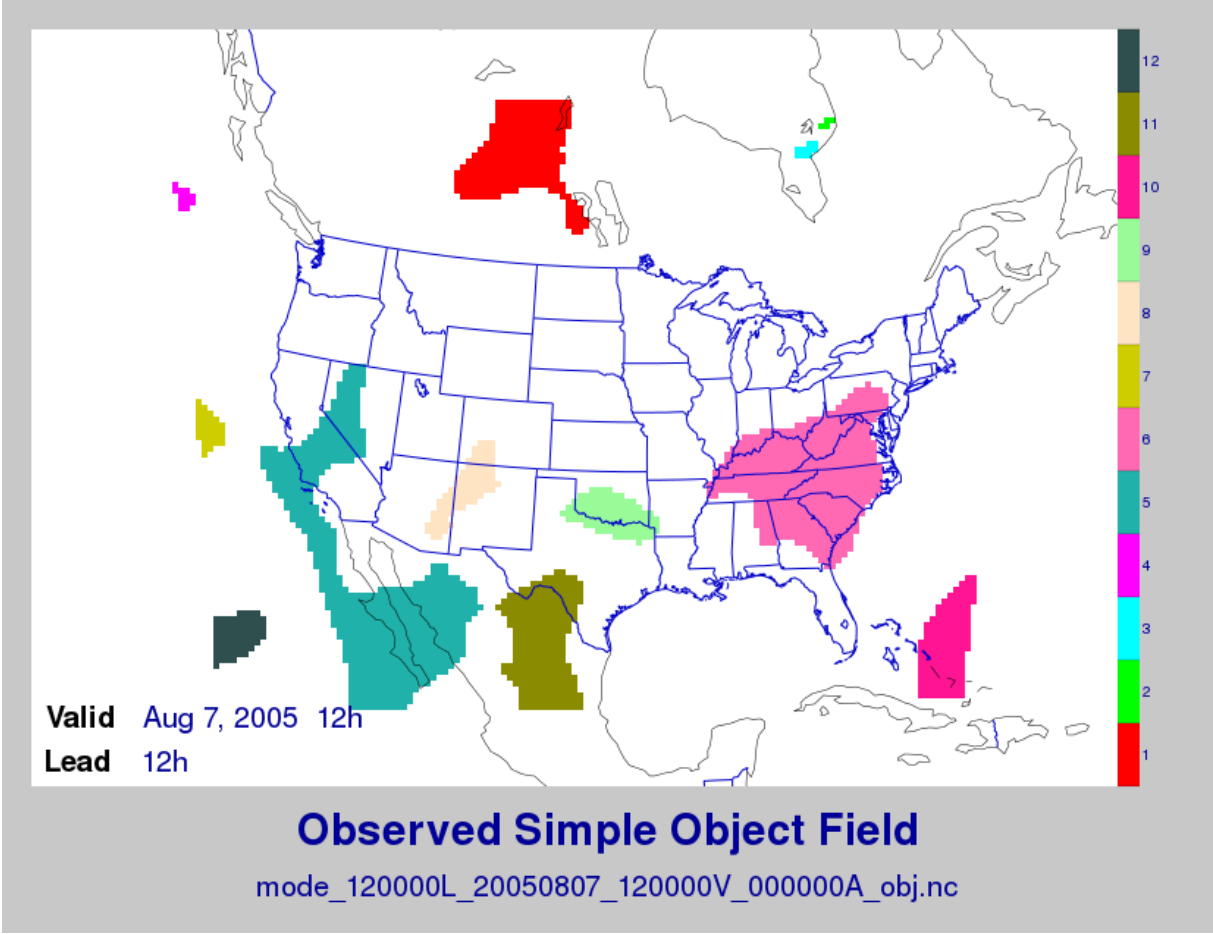

Figure 25.1: Simple observed precipitation objects

Once MET has been applied to forecast and observed fields (or observing locations), and the output has been sorted through the Analysis Tool, numerous graphical and summary analyses can be performed depending on a specific user's needs. Here we give some examples of graphics and summary scores that one might wish to compute with the given output of MET and MET-TC. Any computing language could be used for this stage; some scripts will be provided on the MET users web page (`https://dtcenter.org/community-code/model-evaluation-tools-met`) as examples to assist users.

25.2 Examples of plotting MET output

25.2.1 Grid-Stat tool examples

The plots in Figure 25.2 show time series of frequency bias and Gilbert Skill Score, stratified according to time of day. This type of figure is particularly useful for diagnosing problems that are tied to the diurnal cycle. In this case, two of the models (green dash-dotted and black dotted lines) show an especially high Bias (near 3) during the afternoon (15-21 UTC; left panel), while the skill (GSS; right panel) appears to be

best for the models represented by the solid black line and green dashed lines in the morning (09-15 UTC). Note that any judgment of skill based on GSS should be restricted to times when the Bias is close to one.

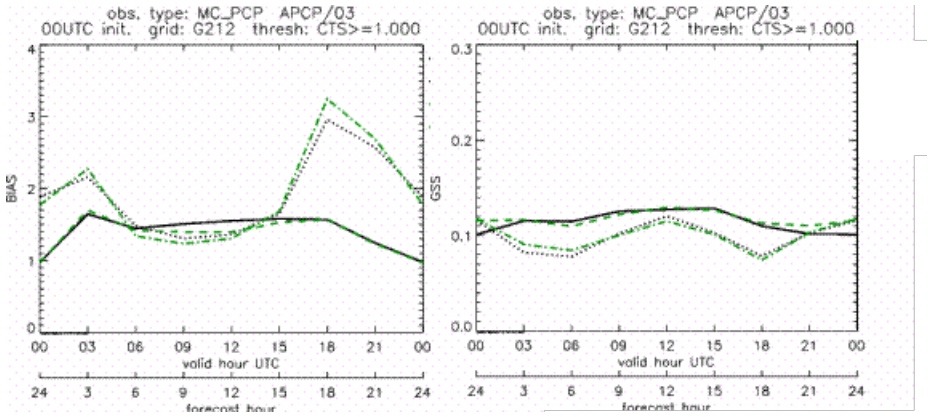

Figure 25.2: Time series of forecast area bias and Gilbert Skill Score for four model configurations (different lines) stratified by time-of-day.

25.2.2 MODE tool examples

When using the MODE tool, it is possible to think of matched objects as hits and unmatched objects as false alarms or misses depending on whether the unmatched object is from the forecast or observed field, respectively. Because the objects can have greatly differing sizes, it is useful to weight the statistics by the areas, which are given in the output as numbers of grid squares. When doing this, it is possible to have different matched observed object areas from matched forecast object areas so that the number of hits will be different depending on which is chosen to be a hit. When comparing multiple forecasts to the same observed field, it is perhaps wise to always use the observed field for the hits so that there is consistency for subsequent comparisons. Defining hits, misses and false alarms in this way allows one to compute many traditional verification scores without the problem of small-scale discrepancies; the matched objects are defined as being matched because they are "close" by the fuzzy logic criteria. Note that scores involving the number of correct negatives may be more difficult to interpret as it is not clear how to define a correct negative in this context. It is also important to evaluate the number and area attributes for these objects in order to provide a more complete picture of how the forecast is performing.

Figure 25.3 gives an example of two traditional verification scores (Bias and CSI) along with bar plots showing the total numbers of objects for the forecast and observed fields, as well as bar plots showing their total areas. These data are from the same set of 13-km WRF model runs analyzed in Figure 25.3. The model runs were initialized at 0 UTC and cover the period 15 July to 15 August 2005. For the forecast evaluation, we compared 3-hour accumulated precipitation for lead times of 3-24 hours to Stage II radar-gauge precipitation. Note that for the 3-hr lead time, indicated as the 0300 UTC valid time in Figure 25.2, the Bias is significantly larger than the other lead times. This is evidenced by the fact that there are both a larger number of forecast objects, and a larger area of forecast objects for this lead time, and only for this lead time. Dashed lines show about 2 bootstrap standard deviations from the estimate.

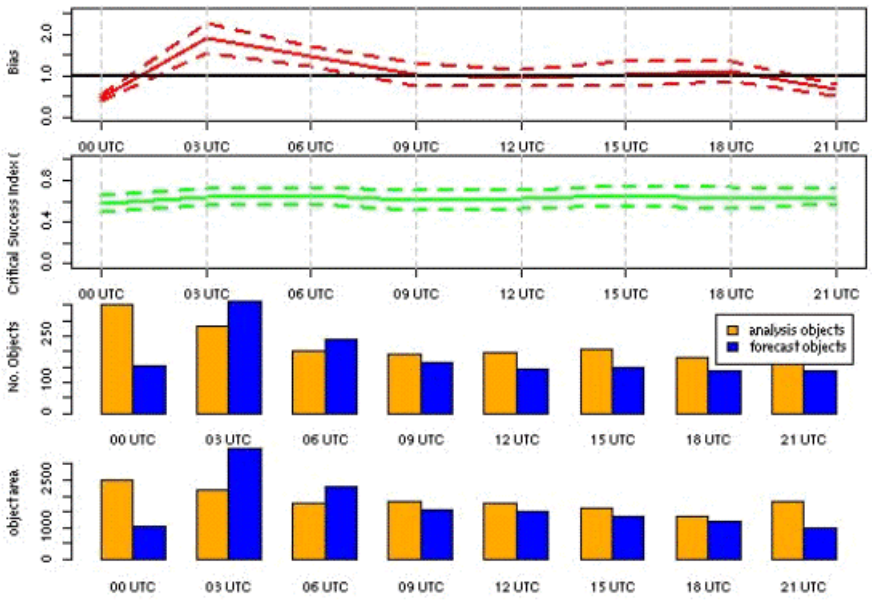

Figure 25.3: Traditional verification scores applied to output of the MODE tool, computed by defining matched observed objects to be hits, unmatched observed objects to be misses, and unmatched forecast objects to be false alarms; weighted by object area. Bar plots show numbers (penultimate row) and areas (bottom row) of observed and forecast objects, respectively.

In addition to the traditional scores, MODE output allows more information to be gleaned about forecast performance. It is even useful when computing the traditional scores to understand how much the forecasts are displaced in terms of both distance and direction. Figure 25.4, for example, shows circle histograms for matched objects. The petals show the percentage of times the forecast object centroids are at a given angle from the observed object centroids. In Figure 25.4 (top diagram) about 25% of the time the forecast object centroids are west of the observed object centroids, whereas in Figure 25.4 (bottom diagram) there is less bias in terms of the forecast objects' centroid locations compared to those of the observed objects, as evidenced by the petals' relatively similar lengths, and their relatively even dispersion around the circle. The colors on the petals represent the proportion of centroid distances within each colored bin along each direction. For example, Figure 25.4 (top row) shows that among the forecast object centroids that are located to the West of the observed object centroids, the greatest proportion of the separation distances (between the observed and forecast object centroids) is greater than 20 grid squares.

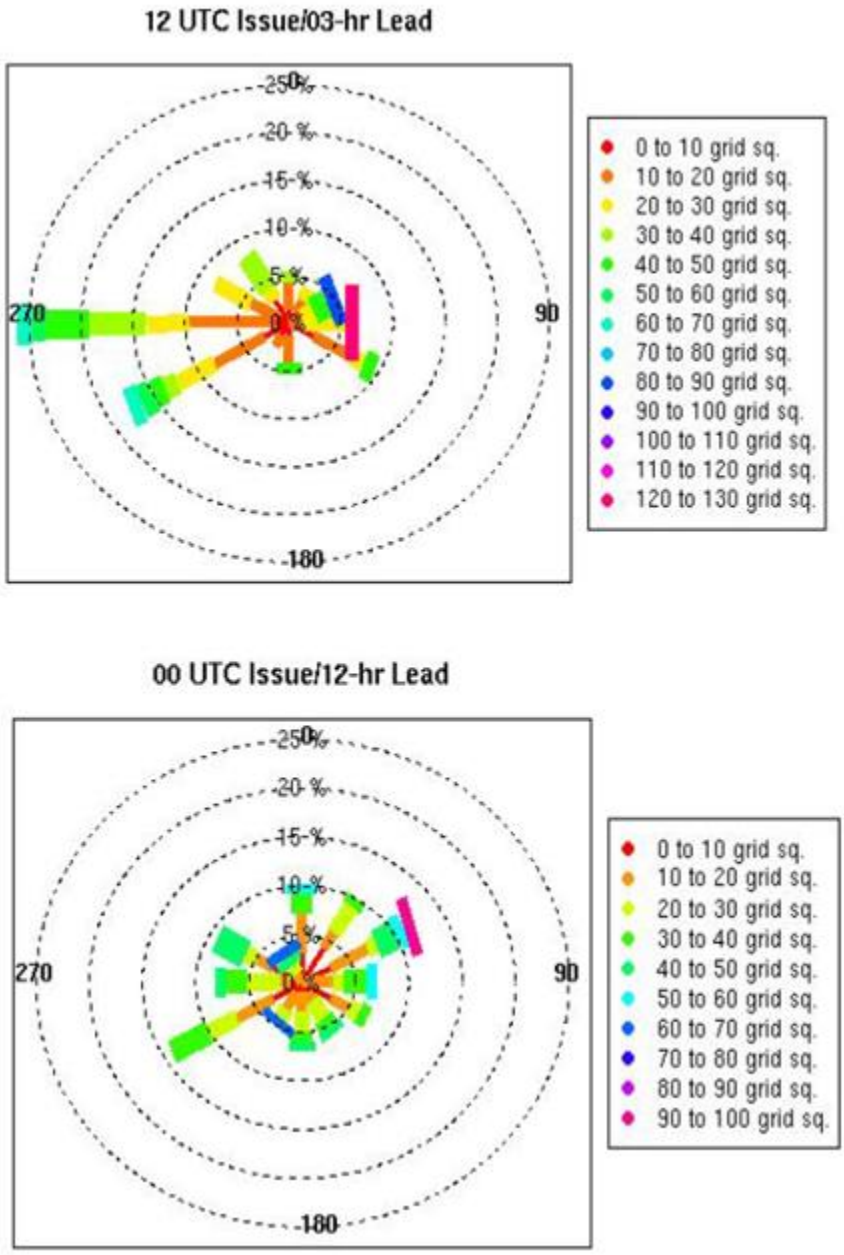

Figure 25.4: Circle histograms showing object centroid angles and distances (see text for explanation).

25.2.3 TC-Stat tool example

There is a basic R script located in the MET installation, share/met/Rscripts/plot_tcmpr.R. The usage statement with a short description of the options for *plot_tcmpr.R* can be obtained by typing: Rscript *plot_tcmpr.R* with no additional arguments. The only required argument is the *-lookin* source, which is the path to the TC-Pairs TCST output files. The R script reads directly from the TC-Pairs output, and calls TC-Stat directly for filter jobs specified in the "*-filter options*" argument.

In order to run this script, the MET_INSTALL_DIR environment variable must be set to the MET installation directory and the MET_BASE environment variable must be set to the MET_INSTALL_DIR/share/met directory. In addition, the *tc_stat tool* under MET_INSTALL_DIR/bin must be in your system path.

The supplied R script can generate a number of different plot types including boxplots, mean, median, rank, and relative performance. Pairwise differences can be plotted for the boxplots, mean, and median. Normal confidence intervals are applied to all figures unless the no_ci option is set to TRUE. Below are two example plots generated from the tools.

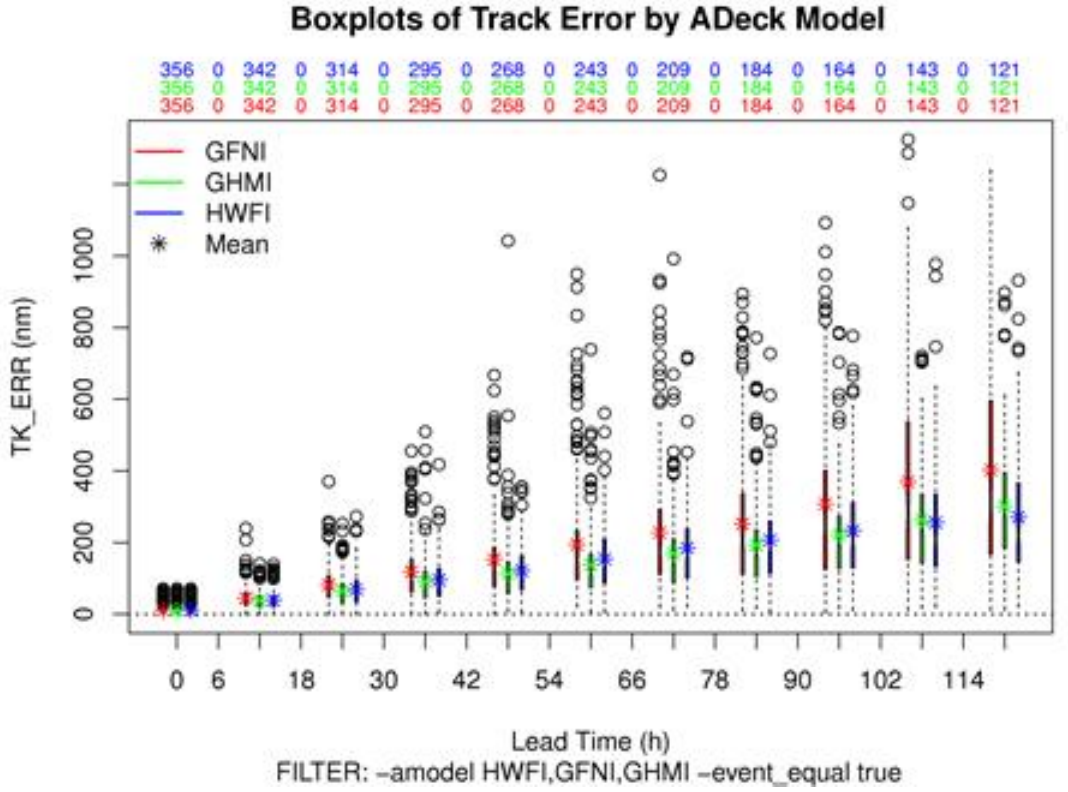

Figure 25.5: Example boxplot from plot_tcmpr.R. Track error distributions by lead time for three operational models GFNI, GHMI, HFWI.

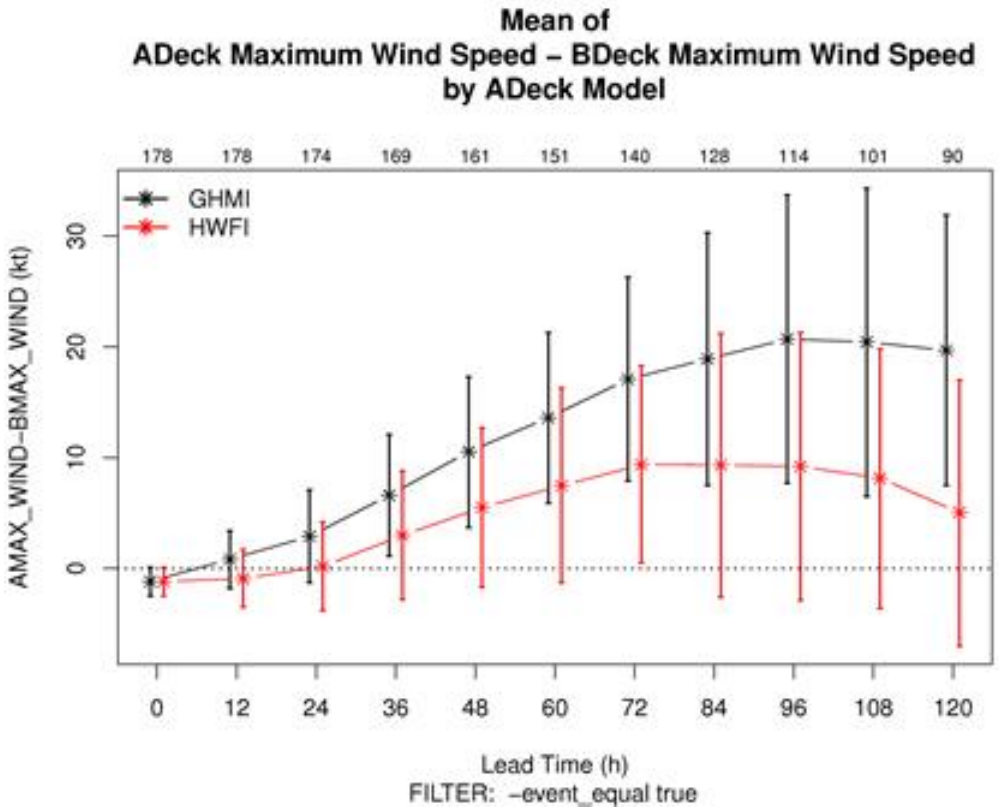

Figure 25.6: Example mean intensity error with confidence intervals at 95% from plot_tcmpr.R. Raw intensity error by lead time for a homogeneous comparison of two operational models GHMI, HWFI.

References

Alberson, S.D., 1998: Five-day Tropical cyclone track forecasts in the North Atlantic Basin. *Weather & Forecasting*, 13, 1005-1015.

Barker, T. W., 1991: The relationship between spread and forecast error in extended-range forecasts. *J. Climate*, 4, 733–742.

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.

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, 1748–1754.

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.`

Buizza, R., 1997: Potential forecast skill of ensemble prediction and spread and skill distributions of the ECMWF ensemble prediction system. Mon. Wea. Rev.,125, 99–119.

Bullock, R., T. Fowler, and B. Brown, 2016: Method for Object-Based Diagnostic Evaluation. NCAR Tech. Note NCAR/TN-532+STR, 66 pp.

Candille, G., and O. Talagrand, 2008: Impact of observational error on the validation of ensemble prediction systems. *Q. J. R. Meteorol. Soc.* 134: 959–971.

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.

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.

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.

Dawid, A.P., 1984: Statistical theory: The prequential approach. *J. Roy. Stat. Soc*, A147, 278-292.

Ebert, E.E., 2008: Fuzzy verification of high-resolution gridded forecasts: a review and proposed framework. *Meteorological Applications*, 15, 51-64.

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`

Efron, B. 2007: Correlation and large-scale significance testing. *Journal of the American Statistical Association*, 102(477), 93-103.

Gilleland, E., 2010: Confidence intervals for forecast verification. *NCAR Technical Note* NCAR/TN-479+STR, 71pp.

Gilleland, E., 2017: A new characterization in the spatial verification framework for false alarms, misses, and overall patterns. *Weather Forecast.*, 32 (1), 187 - 198, doi: 10.1175/WAF-D-16-0134.1.

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/`]

Hamill, T. M., 2001: Interpretation of rank histograms for verifying ensemble forecasts. *Mon. Wea. Rev.*, 129, 550-560.

Hersbach, H., 2000: Decomposition of the continuous ranked probability score for ensemble prediction systems. *Wea. Forecasting*, 15, 559–570.

Hogan, R., E. O'Connor, and A. Illingworth, 2009: Verification of cloud-fraction forecasts. *Quart. Jour. Roy. Meteorol. Soc.*, 135, 1494-1511.

Jolliffe, I.T., and D.B. Stephenson, 2012: *Forecast verification. A practitioner's guide in atmospheric science.* Wiley and Sons Ltd, 240 pp.

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.

Mason, S. J., 2004: On Using "Climatology" as a Reference Strategy in the Brier and Ranked Probability Skill Scores. *Mon. Wea. Rev.*, 132, 1891–1895.

Mittermaier, M., 2013: A strategy for verifying near-convection-resolving model forecasts at observing sites. Wea. Forecasting, 29, 185-204.

Mood, A. M., F. A. Graybill and D. C. Boes, 1974: *Introduction to the Theory of Statistics*, McGraw-Hill, 299-338.

Murphy, A.H., and R.L. Winkler, 1987: A general framework for forecast verification. *Monthly Weather Review*, 115, 1330-1338.

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.

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: 1487–1501.

Santos C. and A. Ghelli, 2012: Observational probability method to assess ensemble precipitation forecasts. *Q. J. R. Meteorol. Soc.* 138: 209–221.

Stephenson, D.B., 2000: Use of the "Odds Ratio" for diagnosing forecast skill. *Weather and Forecasting*, 15, 221-232.

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.

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.

Wilks, D.S. 2010: Sampling distributions of the Brier score and Brier skill score under serial dependence. Q.J.R. Meteorol. Soc., 136, 2109–2118. doi:10.1002/qj.709

Wilks, D., 2011: *Statistical methods in the atmospheric sciences.* Elsevier, San Diego.

List of Tables

List of Figures

Appendix A

FAQs & How do I ... ?

A.1 Frequently Asked Questions

Q. Why was the MET written largely in C++ instead of FORTRAN?

A. MET relies upon the object-oriented aspects of C++, particularly in using the MODE tool. Due to time and budget constraints, it also makes use of a pre-existing forecast verification library that was developed at NCAR.

Q. Why is PrepBUFR **used?**

A. The first goal of MET was to replicate the capabilities of existing verification packages and make these capabilities available to both the DTC and the public.

Q. Why is GRIB used?

A. Forecast data from both WRF cores can be processed into GRIB format, and it is a commonly accepted output format for many NWP models.

Q. Is GRIB2 supported?

A. Yes, forecast output in GRIB2 format can be read by MET. Be sure to compile the GRIB2 code by setting the appropriate configuration file options (see Chapter 2).

Q. How does MET differ from the previously mentioned existing verification packages?

A. MET is an actively maintained, evolving software package that is being made freely available to the public through controlled version releases.

Q. How does the MODE tool differ from the Grid-Stat tool?

A. They offer different ways of viewing verification. The Grid-Stat tool provides traditional verification statistics, while MODE provides specialized spatial statistics.

Q. Will the MET work on data in native model coordinates?

A. No - it will not. In the future, we may add options to allow additional model grid coordinate systems.

Q. How do I get help if my questions are not answered in the User's Guide?

A. First, look on our website `https://dtcenter.org/community-code/model-evaluation-tools-met`. If that doesn't answer your question, then email: met_help@ucar.edu.

Q. Where are the graphics?

A. Currently, very few graphics are included. The plotting tools (plot_point_obs, plot_data_plane, and plot_mode_field) can help you visualize your raw data. Also, ncview can be used with the NetCDF output from MET tools to visualize results. Further graphics support will be made available in the future on the MET website.

Q. How do I find the version of the tool I am using?

A. Type the name of the tool followed by **-version**. For example, type "**pb2nc -version**".

Q. What are MET's conventions for latitude, longitude, azimuth and bearing angles?

A. MET considers north latitude and east longitude positive. Latitudes have range from $-90°$ to $+90°$. Longitudes have range from $-180°$ to $+180°$. Plane angles such as azimuths and bearing (example: horizontal wind direction) have range $0°$ to $360°$ and are measured clockwise from north.

A.2 Troubleshooting

The first place to look for help with individual commands is this user's guide or the usage statements that are provided with the tools. Usage statements for the individual MET tools are available by simply typing the name of the executable in MET's **bin**/ directory. Example scripts available in the MET's **scripts**/ directory show examples of how one might use these commands on example datasets. Here are suggestions on other things to check if you are having problems installing or running MET.

MET won't compile

* Have you specified the locations of NetCDF, GNU Scientific Library, and BUFRLIB, and optional additional libraries using corresponding MET_ environment variables prior to running configure?

* Have these libraries been compiled and installed using the same set of compilers used to build MET?

* Are you using NetCDF version 3.4 or version 4? Currently, only NetCDF version 3.6 can be used with MET.

Grid_stat won't run

* Are both the observational and forecast datasets on the same grid?

MODE won't run

* If using precipitation, do you have the same accumulation periods for both the forecast and observations? (If you aren't sure, run pcp_combine.)
* Are both the observation and forecast datasets on the same grid?

Point-Stat won't run

* Have you run pb2nc first on your PrepBUFR observation data?

Error while loading shared libraries

* Add the lib dir to your LD_LIBRARY_PATH. For example, if you receive the following error: "./mode_analysis: error while loading shared libraries: libgsl.so.19: cannot open shared object file: No such file or directory", you should add the path to the gsl lib (for example, /home/user/MET/gsl-2.1/lib) to your LD_LIBRARY_PATH.

General troubleshooting

* For configuration files used, make certain to use empty square brackets (e.g. [ ]) to indicate no stratification is desired. Do NOT use empty double quotation marks inside square brackets (e.g. ["]).
* Have you designated all the required command line arguments?
* Try rerunning with a higher verbosity level. Increasing the verbosity level to 4 or 5 prints much more diagnostic information to the screen.

A.3 Where to get help

If none of the above suggestions have helped solve your problem, help is available through: met_help@ucar.edu

A.4 How to contribute code

If you have code you would like to contribute, we will gladly consider your contribution. Please send email to: met_help@ucar.edu

Appendix B

Map Projections, Grids, and Polylines

B.1 Map Projections

The following map projections are currently supported in MET:

* Lambert Conformal Projection

* Polar Stereographic Projection (Northern)

* Polar Stereographic Projection (Southern)

* Mercator Projection

* Lat/Lon Projection

* Rotated Lat/Lon Projection

B.2 Grids

All of NCEP's pre-defined grids that reside on one of the projections listed above are implemented in MET. The user may specify one of these NCEP grids in the configuration files as "GNNN" where NNN is the 3-digit NCEP grid number. Defining a new masking grid in MET would involve modifying the vx_data_grids library and recompiling.

Please see NCEP's website for a description and plot of these pre-defined grids:

`http://www.nco.ncep.noaa.gov/pmb/docs/on388/tableb.html`.

B.3 Polylines for NCEP Regions

Many of NCEP's pre-defined verification regions are implemented in MET as lat/lon polyline files. The user may specify one of these NCEP verification regions in the configuration files by pointing to the lat/lon polyline file in the installed share/met/poly directory. Users may also easily define their own lat/lon polyline files.

See NCEP's website for a description and plot of these pre-defined verification regions: `http://www.emc.ncep.noaa.gov/mmb/research/nearsfc/nearsfc.verf.html`

The NCEP verification regions that are implemented in MET as lat/lon polylines are listed below:

* APL.poly for the Appalachians

* ATC.poly for the Arctic Region

* CAM.poly for Central America

* CAR.poly for the Caribbean Sea

* ECA.poly for Eastern Canada

* GLF.poly for the Gulf of Mexico

* GMC.poly for the Gulf of Mexico Coast

* GRB.poly for the Great Basin

* HWI.poly for Hawaii

* LMV.poly for the Lower Mississippi Valley

* MDW.poly for the Midwest

* MEX.poly for Mexico

* NAK.poly for Northern Alaska

* NAO.poly for Northern Atlantic Ocean

* NEC.poly for the Northern East Coast

* NMT.poly for the Northern Mountain Region

* NPL.poly for the Northern Plains

* NPO.poly for the Northern Pacific Ocean

* NSA.poly for Northern South America

* NWC.poly for Northern West Coast

* PRI.poly for Puerto Rico and Islands

* SAK.poly for Southern Alaska

* SAO.poly for the Southern Atlantic Ocean

* SEC.poly for the Southern East Coast

* SMT.poly for the Southern Mountain Region

* SPL.poly for the Southern Plains

* SPO.poly for the Southern Pacific Ocean

* SWC.poly for the Southern West Coast

* SWD.poly for the Southwest Desert

* WCA.poly for Western Canada

* EAST.poly for the Eastern United States (consisting of APL, GMC, LMV, MDW, NEC, and SEC)

* WEST.poly for the Western United States (consisting of GRB, NMT, NPL, NWC, SMT, SPL, SWC, and SWD)

* CONUS.poly for the Continental United States (consisting of EAST and WEST)

Appendix C

Verification Measures

This appendix provides specific information about the many verification statistics and measures that are computed by MET. These measures are categorized into measures for categorical (dichotomous) variables; measures for continuous variables; measures for probabilistic forecasts and measures for neighborhood methods. While the continuous, categorical, and probabilistic statistics are computed by both the Point-Stat and Grid-Stat tools, the neighborhood verification measures are only provided by the Grid-Stat tool.

C.1 MET verification measures for categorical (dichotomous) variables

The verification statistics for dichotomous variables are formulated using a contingency table such as the one shown in Table C-1. In this table f represents the forecasts and o represents the observations; the two possible forecast and observation values are represented by the values 0 and 1. The values in Table C-1 are counts of the number of occurrences of the four possible combinations of forecasts and observations.

Table C.1: 2x2 contingency table in terms of counts. The $\mathbf{n_{ij}}$ values in the table represent the counts in each forecast-observation category, where **i** represents the forecast and **j** represents the observations. The "." symbols in the total cells represent sums across categories.

Forecast	Observation		Total
	o = 1 (e.g., "Yes")	o = 0 (e.g., "No")	
f = 1 (e.g., "Yes")	$\mathbf{n}_{11}$	$\mathbf{n}_{10}$	$\mathbf{n}_{1.} = \mathbf{n}_{11} + \mathbf{n}_{10}$
f = 0 (e.g., "No")	$\mathbf{n}_{01}$	$\mathbf{n}_{00}$	$\mathbf{n}_{0.} = \mathbf{n}_{01} + \mathbf{n}_{00}$
Total	$\mathbf{n}_{.1} = \mathbf{n}_{11} + \mathbf{n}_{01}$	$\mathbf{n}_{.0} = \mathbf{n}_{10} + \mathbf{n}_{00}$	$\mathbf{T} = \mathbf{n}_{11} + \mathbf{n}_{10} + \mathbf{n}_{01} + \mathbf{n}_{00}$

The counts, n_{11}, n_{10}, n_{01}, and n_{00}, are sometimes called the "Hits", "False alarms", "Misses", and "Correct rejections", respectively.

By dividing the counts in the cells by the overall total, T, the joint proportions, $\mathbf{p}_{11}$, $\mathbf{p}_{10}$, $\mathbf{p}_{01}$, and $\mathbf{p}_{00}$ can be computed. Note that $\mathbf{p}_{11} + \mathbf{p}_{10} + \mathbf{p}_{01} + \mathbf{p}_{00} = \mathbf{1}$. Similarly, if the counts are divided by the

row (column) totals, conditional proportions, based on the forecasts (observations) can be computed. All of these combinations and the basic counts can be produced by the Point-Stat tool.

The values in Table C-1 can also be used to compute the F, O, and H relative frequencies that are produced by the NCEP Verification System, and the Point-Stat tool provides an option to produce the statistics in this form. In terms of the other statistics computed by the Point-Stat tool, F is equivalent to the Mean Forecast; H is equivalent to POD; and O is equivalent to the Base Rate. All of these statistics are defined in the subsections below. The Point-Stat tool also provides the total number of observations, **T**.

The categorical verification measures produced by the Point-Stat and Grid-Stat tools are described in the following subsections. They are presented in the order shown in 7.2 through 7.5.

TOTAL

The total number of forecast-observation pairs, **T**.

Base rate

Called "O_RATE" in FHO output (7.2)

Called "BASER" in CTS output (7.4)

The base rate is defined as $\bar{o} = \frac{n_{11} + n_{01}}{T} = \frac{n_{.1}}{T}$. This value is also known as the sample climatology, and is the relative frequency of occurrence of the event (i.e., $\mathbf{o} = \mathbf{1}$). The base rate is equivalent to the "O" value produced by the NCEP Verification System.

Mean forecast

Called "F_RATE" in FHO output (7.2)

Called "FMEAN" in CTS output (7.4)

The mean forecast value is defined as $\overline{f} = \frac{n_{11} + n_{10}}{T} = \frac{n_{1.}}{T}$.

This statistic is comparable to the base rate and is the relative frequency of occurrence of a forecast of the event (i.e., $\mathbf{f} = \mathbf{1}$). The mean forecast is equivalent to the "F" value computed by the NCEP Verification System.

Accuracy

Called "ACC" in CTS output (7.4)

Accuracy for a 2x2 contingency table is defined as

$$\text{ACC} = \frac{n_{11} + n_{00}}{T}.$$

That is, it is the proportion of forecasts that were either hits or correct rejections - the fraction that were correct. Accuracy ranges from 0 to 1; a perfect forecast would have an accuracy value of 1. Accuracy should be used with caution, especially for rare events, because it can be strongly influenced by large values of $\mathbf{n_{00}}$.

Frequency Bias

Called "FBIAS" in CTS output (7.4)

Frequency Bias is the ratio of the total number of forecasts of an event to the total number of observations of the event. It is defined as

$$\text{Bias} = \frac{n_{11} + n_{10}}{n_{11} + n_{01}} = \frac{n_{1\cdot}}{n_1}.$$

A "good" value of Frequency Bias is close to 1; a value greater than 1 indicates the event was forecasted too frequently and a value less than 1 indicates the event was not forecasted frequently enough.

Probability of Detection (POD)

Called "H_RATE" in FHO output (7.2)

Called "PODY" in CTS output (7.4)

POD is defined as

$$\text{POD} = \frac{n_{11}}{n_{11} + n_{01}} = \frac{n_{11}}{n_1}.$$

It is the fraction of events that were correctly forecasted to occur. POD is equivalent to the H value computed by the NCEP verification system and is also known as the hit rate. POD ranges from 0 to 1; a perfect forecast would have POD = 1.

Probability of False Detection (POFD)

Called "POFD" in CTS output (7.4)

POFD is defined as

$$\text{POFD} = \frac{n_{10}}{n_{10} + n_{00}} = \frac{n_{10}}{n_{.0}}.$$

It is the proportion of non-events that were forecast to be events. POFD is also often called the False Alarm Rate. POFD ranges from 0 to 1; a perfect forecast would have POFD = 0.

Probability of Detection of the non-event (PODn)

Called "PODN" in CTS output (7.4)

PODn is defined as

$$\text{PODN} = \frac{n_{00}}{n_{10} + n_{00}} = \frac{n_{00}}{n_{.0}}.$$

It is the proportion of non-events that were correctly forecasted to be non-events. Note that PODn = 1 - POFD. PODn ranges from 0 to 1. Like POD, a perfect forecast would have PODn = 1.

False Alarm Ratio (FAR)

Called "FAR" in CTS output (7.4)

FAR is defined as

$$\text{FAR} = \frac{n_{10}}{n_{10} + n_{11}} = \frac{n_{10}}{n_{1.}}.$$

It is the proportion of forecasts of the event occurring for which the event did not occur. FAR ranges from 0 to 1; a perfect forecast would have FAR = 0.

Critical Success Index (CSI)

Called "CSI" in CTS output (7.4)

CSI is defined as

$$\text{CSI} = \frac{n_{11}}{n_{11} + n_{10} + n_{01}}.$$

It is the ratio of the number of times the event was correctly forecasted to occur to the number of times it was either forecasted or occurred. CSI ignores the "correct rejections" category (i.e., $\mathbf{n_{00}}$). CSI is also known as the Threat Score (TS). CSI can also be written as a nonlinear combination of POD and FAR, and is strongly related to Frequency Bias and the Base Rate.

Gilbert Skill Score (GSS)

Called "GSS" in CTS output (7.4)

GSS is based on the CSI, corrected for the number of hits that would be expected by chance. In particular,

$$\text{GSS} = \frac{n_{11} - C_1}{n_{11} + n_{10} + n_{01} - C_1},$$

where

$$C = \frac{(n_{11} + n_{10})(n_{11} + n_{01})}{T}.$$

GSS is also known as the Equitable Threat Score (ETS). GSS values range from -1/3 to 1. A no-skill forecast would have GSS = 0; a perfect forecast would have GSS = 1.

Hanssen-Kuipers Discriminant (H-K)

Called "HK" in CTS output (7.4)

H-K is defined as

$$\text{H-K} = \frac{n_{11}n_{00} - n_{10}n_{01}}{(n_{11} + n_{01})(n_{10} + n_{00})}.$$

More simply, H-K = POD − POFD.

H-K is also known as the True Skill Statistic (TSS) and less commonly (although perhaps more properly) as the Peirce Skill Score. H-K measures the ability of the forecast to discriminate between (or correctly classify) events and non-events. H-K values range between -1 and 1. A value of 0 indicates no skill; a perfect forecast would have H-K = 1.

Heidke Skill Score (HSS)

Called "HSS" in CTS output (7.4)

HSS is a skill score based on Accuracy, where the Accuracy is corrected by the number of correct forecasts that would be expected by chance. In particular,

$$\text{HSS} = \frac{n_{11} + n_{00} - C_2}{T - C_2},$$

where

$$C_2 = \frac{(n_{11} + n_{10})(n_{11} + n_{01}) + (n_{01} + n_{00})(n_{10} + n_{00})}{T}.$$

HSS can range from minus infinity to 1. A perfect forecast would have HSS = 1.

Odds Ratio (OR)

Called "ODDS" in CTS output (7.4)

OR measures the ratio of the odds of a forecast of the event being correct to the odds of a forecast of the event being wrong. OR is defined as

$$\text{OR} = \frac{n_{11} \times n_{00}}{n_{10} \times n_{01}} = \frac{\left(\frac{\text{POD}}{\text{1-POD}}\right)}{\left(\frac{\text{POFD}}{\text{1-POFD}}\right)}.$$

OR can range from 0 to ∞. A perfect forecast would have a value of OR = infinity. OR is often expressed as the log Odds Ratio or as the Odds Ratio Skill Score (Stephenson 2000).

Logarithm of the Odds Ratio (LODDS)

Called "LODDS" in CTS output (7.4)

LODDS transforms the odds ratio via the logarithm, which tends to normalize the statistic for rare events (Stephenson 2000). However, it can take values of $\pm\infty$when any of the contingency table counts is 0. LODDS is defined as LODDS = $ln(OR)$.

Odds Ratio Skill Score (ORSS)

Called "ORSS" in CTS output (7.4)

ORSS is a skill score based on the odds ratio. ORSS is defined as

$$\text{ORSS} = \frac{OR - 1}{OR + 1}.$$

ORSS is sometime also referred to as Yule's Q. (Stephenson 2000).

Extreme Dependency Score (EDS)

Called "EDS" in CTS output (7.4)

The extreme dependency score measures the association between forecast and observed rare events. EDS is defined as

$$\text{EDS} = \frac{2\ln\left(\frac{n_{11}+n_{01}}{T}\right)}{\ln\left(\frac{n_{11}}{T}\right)} - 1.$$

EDS can range from -1 to 1, with 0 representing no skill. A perfect forecast would have a value of EDS = 1. EDS is independent of bias, so should be presented along with the frequency bias statistic (Stephenson et al, 2008).

Extreme Dependency Index (EDI)

Called "EDI" in CTS output (7.4)

The extreme dependency index measures the association between forecast and observed rare events. EDI is defined as , where and are the Hit Rate and False Alarm Rate, respectively.

EDI can range from -∞ to 1, with 0 representing no skill. A perfect forecast would have a value of EDI = 1 (Ferro and Stephenson, 2011).

Symmetric Extreme Dependency Score (SEDS)

Called "SEDS" in CTS output (7.4)

The symmetric extreme dependency score measures the association between forecast and observed rare events. SEDS is defined as

$$\text{SEDS} = \frac{2\ln\left[\frac{(n_{11}+n_{01})(n_{11}+n_{10})}{T^2}\right]}{\ln\left(\frac{n_{11}}{T}\right)} - 1.$$

SEDS can range from -∞ to 1, with 0 representing no skill. A perfect forecast would have a value of SEDS = 1 (Ferro and Stephenson, 2011).

Symmetric Extremal Dependency Index (SEDI)

Called "SEDI" in CTS output (7.4)

The symmetric extremal dependency index measures the association between forecast and observed rare events. SEDI is defined as

$$\text{SEDI} = \frac{\ln F - \ln H + \ln(1-H) - \ln(1-F)}{\ln F + \ln H + \ln(1-H) + \ln(1-F)},$$

where $H = \dfrac{n_{11}}{n_{11}+n_{01}}$ and $F = \dfrac{n_{10}}{n_{00}+n_{10}}$ are the Hit Rate and False Alarm Rate, respectively.

SEDI can range from -∞ to 1, with 0 representing no skill. A perfect forecast would have a value of SEDI = 1. SEDI approaches 1 only as the forecast approaches perfection (Ferro and Stephenson, 2011).

Bias Adjusted Gilbert Skill Score (GSS)

Called "BAGSS" in CTS output (7.4)

BAGSS is based on the GSS, but is corrected as much as possible for forecast bias (Brill and Mesinger, 2009).

Economic Cost Loss Relative Value (ECLV)

Included in ECLV output (7.14)

The Economic Cost Loss Relative Value (ECLV) applies a weighting to the contingency table counts to determine the relative value of a forecast based on user-specific information. The cost is incurred to protect against an undesirable outcome, whether that outcome occurs or not. No cost is incurred if no protection is undertaken. Then, if the event occurs, the user sustains a loss. If the event does not occur, there is neither a cost nor a loss. The maximum forecast value is achieved when the cost/loss ratio equals the climatological probability. When this occurs, the ECLV is equal to the Hanssen and Kuipers discriminant. The Economic Cost Loss Relative Value is defined differently depending on whether the cost / loss ratio is lower than the base rate or higher. The ECLV is a function of the cost / loss ratio (cl), the hit rate (h), the false alarm rate (f), the miss rate (m), and the base rate (b).

For cost / loss ratio below the base rate, the ECLV is defined as:

$$\text{ECLV} = \frac{(cl * (h + f - 1)) + m}{cl * (b - 1)}.$$

For cost / loss ratio above the base rate, the ECLV is defined as:

$$\text{ECLV} = \frac{(cl * (h + f)) + m - b}{b * (cl - 1)}.$$

C.2 MET verification measures for continuous variables

For continuous variables, many verification measures are based on the forecast error (i.e., **f - o**). However, it also is of interest to investigate characteristics of the forecasts, and the observations, as well as their relationship. These concepts are consistent with the general framework for verification outlined by Murphy and Winkler (1987). The statistics produced by MET for continuous forecasts represent this philosophy of verification, which focuses on a variety of aspects of performance rather than a single measure.

The verification measures currently evaluated by the Point-Stat tool are defined and described in the subsections below. In these definitions, **f** represents the forecasts, **o** represents the observation, and **n** is the number of forecast-observation pairs.

Mean forecast

Called "FBAR" in CNT output (7.6)

Called "FBAR" in SL1L2 output (7.15)

The sample mean forecast, FBAR, is defined as $\overline{f} = \frac{1}{n}\sum_{i=1}^{n} f_i$.

Mean observation

Called "OBAR" in CNT output (7.6)

Called "OBAR" in SL1L2 output (7.15)

The sample mean observation is defined as $\overline{o} = \frac{1}{n}\sum_{i=1}^{n} o_i$.

Forecast standard deviation

Called "FSTDEV" in CNT output (7.6)

The sample variance of the forecasts is defined as

$$s_f^2 = \frac{1}{T-1}\sum_{i=1}^{T}(f_i - \overline{f})^2.$$

The forecast standard deviation is defined as $s_f = \sqrt{s_f^2}$.

Observation standard deviation

Called "OSTDEV" in CNT output (7.6)

The sample variance of the observations is defined as

$$s_o^2 = \frac{1}{T-1}\sum_{i=1}^{T}(o_i - \overline{o})^2.$$

The observed standard deviation is defined as $s_o = \sqrt{s_o^2}$.

Pearson Correlation Coefficient

Called "PR_CORR" in CNT output (7.6)

The Pearson correlation coefficient, **r**, measures the strength of linear association between the forecasts and observations. The Pearson correlation coefficient is defined as:

$$r = \frac{\sum_{i=1}^{T}(f_i - \overline{f})(o_i - \overline{o})}{\sqrt{\sum(f_i - \overline{f})^2}\sqrt{\sum(o_i - \overline{o})^2}}$$

r can range between -1 and 1; a value of 1 indicates perfect correlation and a value of -1 indicates perfect negative correlation. A value of 0 indicates that the forecasts and observations are not correlated.

Spearman rank correlation coefficient (ρ_s)

Called "SP_CORR" in CNT (7.6)

The Spearman rank correlation coefficient (ρ_s) is a robust measure of association that is based on the ranks of the forecast and observed values rather than the actual values. That is, the forecast and observed samples are ordered from smallest to largest and rank values (from 1 to **n**, where **n** is the total number of pairs) are assigned. The pairs of forecast-observed ranks are then used to compute a correlation coefficient, analogous to the Pearson correlation coefficient, **r**.

A simpler formulation of the Spearman-rank correlation is based on differences between the each of the pairs of ranks (denoted as d_i):

$$\rho_s = \frac{6}{n(n^2-1)}\sum_{i=1}^{n} d_i^2$$

Like **r**, the Spearman rank correlation coefficient ranges between -1 and 1; a value of 1 indicates perfect correlation and a value of -1 indicates perfect negative correlation. A value of 0 indicates that the forecasts and observations are not correlated.

Kendall's Tau statistic (τ)

Called "KT_CORR" in CNT output (7.6)

Kendall's Tau statistic (τ) is a robust measure of the level of association between the forecast and observation pairs. It is defined as
$\tau = \frac{N_c - N_p}{n(n-1)/2}$.

where NC is the number of "concordant" pairs and ND is the number of "discordant" pairs. Concordant pairs are identified by comparing each pair with all other pairs in the sample; this can be done most easily

by ordering all of the (f_i, o_i) pairs according to f_i, in which case the o_i values won't necessarily be in order. The number of concordant matches of a particular pair with other pairs is computed by counting the number of pairs (with larger values) for which the value of oi for the current pair is exceeded (that is, pairs for which the values of **f** and **o** are both larger than the value for the current pair). Once this is done, Nc is computed by summing the counts for all pairs. The total number of possible pairs is ; thus, the number of discordant pairs is .

Like **r** and ρ_s, Kendall's Tau (τ) ranges between -1 and 1; a value of 1 indicates perfect association (concordance) and a value of -1 indicates perfect negative association. A value of 0 indicates that the forecasts and observations are not associated.

Mean Error (ME)

Called "ME" in CNT output (7.6)

The Mean Error, ME, is a measure of overall bias for continuous variables; in particular ME = Bias. It is defined as
ME = $\frac{1}{n}\sum_{i=1}^{n}(f_i - o_i) = \overline{f} - \overline{o}$.

A perfect forecast has ME = 0.

Mean Error Squared (ME2)

Called "ME2" in CNT output (7.6)

The Mean Error Squared, ME2, is provided to give a complete breakdown of MSE in terms of squared Bias plus estimated variance of the error, as detailed below in the section on BCMSE. It is defined as ME2 = ME^2.

A perfect forecast has ME2 = 0.

Multiplicative Bias

Called "MBIAS" in CNT output (7.6)

Multiplicative bias is simply the ratio of the means of the forecasts and the observations: MBIAS = $\overline{f}/\overline{o}$

Mean-squared error (MSE)

Called "MSE" in CNT output (7.6)

MSE measures the average squared error of the forecasts. Specifically, $MSE = \frac{1}{n}\sum(f_i - o_i)^2$.

Root-mean-squared error (RMSE)

Called "RMSE" in CNT output (7.6)

RMSE is simply the square root of the MSE, $RMSE = \sqrt{MSE}$.

Standard deviation of the error

Called "ESTDEV" in CNT output (7.6)

Bias-Corrected MSE

Called "BCMSE" in CNT output (7.6)

MSE and RMSE are strongly impacted by large errors. They also are strongly impacted by large bias (ME) values. MSE and RMSE can range from 0 to infinity. A perfect forecast would have MSE = RMSE = 0.

MSE can be re-written as $MSE = (\overline{f} - \overline{o})^2 + s_f^2 + s_o^2 - 2 s_f s_o r_{fo}$, where $\overline{f} - \overline{o} = ME$ and $s_f^2 + s_o^2 - 2 s_f s_o r_{fo}$ is the estimated variance of the error, s_{fo}^2. Thus, $MSE = ME^2 + s_{f-o}^2$. To understand the behavior of MSE, it is important to examine both of the terms of MSE, rather than examining MSE alone. Moreover, MSE can be strongly influenced by ME, as shown by this decomposition.

The standard deviation of the error, s_{f-o}, is $s_{f-o} = \sqrt{s_{f-o}^2} = \sqrt{s_f^2 + s_o^2 - 2 s_f s_o r_{fo}}$.

Note that the square of the standard deviation of the error (ESTDEV2) is sometimes called the "Bias-corrected MSE" (BCMSE) because it removes the effect of overall bias from the forecast-observation squared differences.

Mean Absolute Error (MAE)

Called "MAE" in CNT output (7.6)

The Mean Absolute Error (MAE) is defined as $MAE = \frac{1}{n} \sum |f_i - o_i|$.

MAE is less influenced by large errors and also does not depend on the mean error. A perfect forecast would have MAE = 0.

Inter Quartile Range of the Errors (IQR)

Called "IQR" in CNT output (7.6)

The Inter Quartile Range of the Errors (IQR) is the difference between the 75th and 25th percentiles of the errors. It is defined as IQR = $p_{75}(f_i - o_i) - p_{25}(f_i - o_i)$.

IQR is another estimate of spread, similar to standard error, but is less influenced by large errors and also does not depend on the mean error. A perfect forecast would have IQR = 0.

Median Absolute Deviation (MAD)

Called "MAD" in CNT output (7.6)

The Median Absolute Deviation (MAD) is defined as MAD = median$|f_i - o_i|$.

MAD is an estimate of spread, similar to standard error, but is less influenced by large errors and also does not depend on the mean error. A perfect forecast would have MAD = 0.

Mean Squared Error Skill Score

Called "MSESS" in CNT output (7.6)

The Mean Squared Error Skill Score is one minus the ratio of the forecast MSE to some reference MSE, usually climatology. It is sometimes referred to as Murphy's Mean Squared Error Skill Score.

$$\text{MSESS} = 1 - \frac{\text{MSE}_f}{\text{MSE}_r}$$

Root-mean-squared Forecast Anomaly

Called "RMSFA" in CNT output (7.6)

RMSFA is the square root of the average squared forecast anomaly. Specifically, $RMSFA = \sqrt{\frac{1}{n}\sum (f_i - c_i)^2}$.

Root-mean-squared Observation Anomaly

Called "RMSOA" in CNT output (7.6)

RMSOA is the square root of the average squared observation anomaly. Specifically, $RMSOA = \sqrt{\frac{1}{n}\sum (o_i - c_i)^2}$.

Percentiles of the errors

Called "E10", "E25", "E50", "E75", "E90" in CNT output (7.6)

Percentiles of the errors provide more information about the distribution of errors than can be obtained from the mean and standard deviations of the errors. Percentiles are computed by ordering the errors from smallest to largest and computing the rank location of each percentile in the ordering, and matching the rank to the actual value. Percentiles can also be used to create box plots of the errors. In MET, the 0.10th, 0.25th, 0.50th, 0.75th, and 0.90th quantile values of the errors are computed.

Anomaly Correlation Coefficient

Called "ANOM_CORR" in CNT output (7.6)

The Anomaly correlation coefficient is equivalent to the Pearson correlation coefficient, except that both the forecasts and observations are first adjusted according to a climatology value. The anomaly is the difference between the individual forecast or observation and the typical situation, as measured by a climatology (**c**) of some variety. It measures the strength of linear association between the forecast anomalies and observed anomalies. The Anomaly correlation coefficient is defined as:

$$\text{Anomoly Correlation} = \frac{\sum(f_i - c)(o_i - c)}{\sqrt{\sum(f_i - c)^2}\sqrt{\sum(o_i - c)^2}}.$$

Anomaly correlation can range between -1 and 1; a value of 1 indicates perfect correlation and a value of -1 indicates perfect negative correlation. A value of 0 indicates that the forecast and observed anomalies are not correlated.

Partial Sums lines (SL1L2, SAL1L2, VL1L2, VAL1L2) (7.15, 7.16, 7.17, and 7.18)

The SL1L2, SAL1L2, VL1L2, and VAL1L2 line types are used to store data summaries (e.g. partial sums) that can later be accumulated into verification statistics. These are divided according to scalar or vector summaries (S or V). The climate anomaly values (A) can be stored in place of the actuals, which is just a re-centering of the values around the climatological average. L1 and L2 refer to the L1 and L2 norms, the distance metrics commonly referred to as the "city block" and "Euclidean" distances. The city block is the absolute value of a distance while the Euclidean distance is the square root of the squared distance.

The partial sums can be accumulated over individual cases to produce statistics for a longer period without any loss of information because these sums are *sufficient* for resulting statistics such as RMSE, bias, correlation coefficient, and MAE (Mood et al, 1974). Thus, the individual errors need not be stored, all of the information relevant to calculation of statistics are contained in the sums. As an example, the sum of all data points and the sum of all squared data points (or equivalently, the sample mean and sample variance) are *jointly sufficient* for estimates of the Gaussian distribution mean and variance.

Minimally sufficient statistics are those that condense the data most, with no loss of information. Statistics based on L1 and L2 norms allow for good compression of information. Statistics based on other norms, such as order statistics, do not result in good compression of information. For this reason, statistics such as RMSE are often preferred to statistics such as the median absolute deviation. The partial sums are not sufficient for order statistics, such as the median or quartiles.

Scalar L1 and L2 values

Called "FBAR", "OBAR", "FOBAR", "FFBAR", and "OOBAR" in SL1L2 output (7.15)

These statistics are simply the 1st and 2nd moments of the forecasts, observations and errors:

FBAR = Mean(f) = $\overline{f} = \frac{1}{n}\sum_{i=1}^{n} f_i$

OBAR = Mean(o) = $\overline{o} = \frac{1}{n}\sum_{i=1}^{n} o_i$

FOBAR = Mean(fo) = $\overline{fo} = \frac{1}{n}\sum_{i=1}^{n} f_i o_i$

FFBAR = Mean(f^2) = $\overline{f}^2 = \frac{1}{n}\sum_{i=1}^{n} f_i^2$

OOBAR = Mean(o^2) = $\overline{o}^2 = \frac{1}{n}\sum_{i=1}^{n} o_i^2$

Some of the other statistics for continuous forecasts (e.g., RMSE) can be derived from these moments.

Scalar anomaly L1 and L2 values

Called "FABAR", "OABAR", "FOABAR", "FFABAR", "OOABAR" in SAL1L2 output (7.16)

Computation of these statistics requires a climatological value, c. These statistics are the 1st and 2nd moments of the scalar anomalies. The moments are defined as:

FABAR = Mean($f-c$) = $\overline{f-c} = \frac{1}{n}\sum_{i=1}^{n}(f_i - c)$

OABAR = Mean($o-c$) = $\overline{o-c} = \frac{1}{n}\sum_{i=1}^{n}(o_i - c)$

FOABAR = Mean[$(f-c)(o-c)$] = $\overline{(f-c)(o-c)} = \frac{1}{n}\sum_{i=1}^{n}(f_i - c)(o_i - c)$

FFABAR = Mean[$(f-c)^2$] = $\overline{(f-c)}^2 = \frac{1}{n}\sum_{i=1}^{n}(f_i - c)^2$

OOABAR = Mean[$(o-c)^2$] = $\overline{(o-c)}^2 = \frac{1}{n}\sum_{i=1}^{n}(o_i - c)^2$

Vector L1 and L2 values

Called "UFBAR", "VFBAR", "UOBAR", "VOBAR", "UVFOBAR", "UVFFBAR", "UVOOBAR" in VL1L2 output (7.17)

These statistics are the moments for wind vector values, where $\mathbf{u}$ is the E-W wind component and $\mathbf{v}$ is the N-S wind component ($\mathbf{u_f}$ is the forecast E-W wind component; $\mathbf{u_o}$ is the observed E-W wind component; $\mathbf{v_f}$ is the forecast N-S wind component; and $\mathbf{v_o}$ is the observed N-S wind component). The following measures are computed:

UFBAR = Mean(u_f) = $\overline{u}_f = \frac{1}{n}\sum_{i=1}^{n} u_{fi}$

VFBAR = Mean(v_f) = $\overline{v}_f = \frac{1}{n}\sum_{i=1}^{n} v_{fi}$

UOBAR = Mean$(u_o) = \overline{u}_o = \frac{1}{n}\sum_{i=1}^{n} u_{oi}$

VOBAR = Mean$(v_o) = \overline{v}_o = \frac{1}{n}\sum_{i=1}^{n} v_{oi}$

UVFOBAR = Mean$(u_f u_o + v_f v_o) = \frac{1}{n}\sum_{i=1}^{n}(u_{fi}u_{oi} + v_{fi}v_{oi})$

UVFFBAR = Mean$(u_f^2 + v_f^2) = \frac{1}{n}\sum_{i=1}^{n}(u_{fi}^2 + v_{fi}^2)$

UVOOBAR = Mean$(u_o^2 + v_o^2) = \frac{1}{n}\sum_{i=1}^{n}(u_{oi}^2 + v_{oi}^2)$

Vector anomaly L1 and L2 values

Called "UFABAR", "VFABAR", "UOABAR", "VOABAR", "UVFOABAR", "UVFFABAR", "UVOOABAR" in VAL1L2 output (7.18)

These statistics require climatological values for the wind vector components, $\mathbf{u_c}$ and $\mathbf{v_c}$. The measures are defined below:

UFABAR = Mean$(u_f - u_c) = \frac{1}{n}\sum_{i=1}^{n}(u_{fi} - u_c)$

VFBAR = Mean$(v_f - v_c) = \frac{1}{n}\sum_{i=1}^{n}(v_{fi} - v_c)$

UOABAR = Mean$(u_o - u_c) = \frac{1}{n}\sum_{i=1}^{n}(u_{oi} - u_c)$

VOABAR = Mean$(v_o - v_c) = \frac{1}{n}\sum_{i=1}^{n}(v_{oi} - v_c)$

UVFOABAR = Mean$[(u_f - u_c)(u_o - u_c) + (v_f - v_c)(v_o - v_c)] = \frac{1}{n}\sum_{i=1}^{n}(u_{fi} - u_c) + (u_{oi} - u_c) + (v_{fi} - v_c)(v_{oi} - v_c))$

UVFFABAR = Mean$[(u_f - u_c)^2 + (v_f - v_c)^2] = \frac{1}{n}\sum_{i=1}^{n}((u_{fi} - u_c)^2 + (v_{fi} - v_c)^2))$

UVOOABAR = Mean$[(u_o - u_c)^2 + (v_o - v_c)^2] = \frac{1}{n}\sum_{i=1}^{n}((u_{oi} - u_c)^2 + (v_{oi} - v_c)^2))$

Gradient values

Called "TOTAL", "FGBAR", "OGBAR", "MGBAR", "EGBAR", "S1", "S1_OG", and "FGOG_RATIO" in GRAD output (8.6)

These statistics are only computed by the Grid_Stat tool and require vectors. Here ∇ is the gradient operator, which in this applications signifies the difference between adjacent grid points in both the grid-x and grid-y directions. TOTAL is the count of grid locations used in the calculations. The remaining measures are defined below:

FGBAR = Mean$|\nabla f| = \frac{1}{n}\sum_{i=1}^{n}|\nabla f_i|$

OGBAR = Mean$|\nabla o| = \frac{1}{n}\sum_{i=1}^{n}|\nabla o_i|$

$$\text{MGBAR} = \text{Max}(\text{FGBAR}, \text{OGBAR})$$

$$\text{EGBAR} = \text{Mean}|\nabla f - \nabla o| = \frac{1}{n}\sum_{i=1}^{n}|\nabla f_i - \nabla o_i|$$

$$\text{S1} = 100\frac{\sum_{i=1}^{n}(w_i(e_g))}{\sum_{i=1}^{n}(w_i(G_L)_i}},$$

where the weights are applied at each grid location, with values assigned according to the weight option specified in the configuration file. The components of the $S1$ equation are as follows:

$$e_g = \left(\left|\frac{\delta}{\delta x}(f-o)\right| + \left|\frac{\delta}{\delta y}(f-o)\right|\right)$$

$$G_L = \max\left(\left|\frac{\delta f}{\delta x}\right|, \left|\frac{\delta o}{\delta x}\right|\right) + \max\left(\left|\frac{\delta f}{\delta y}\right|, \left|\frac{\delta o}{\delta y}\right|\right)$$

$$\text{S1\_OG} = \frac{\text{EGBAR}}{\text{OGBAR}}$$

$$\text{FGOG\_RATIO} = \frac{\text{FGBAR}}{\text{OGBAR}}$$

C.3 MET verification measures for probabilistic forecasts

The results of the probabilistic verification methods that are included in the Point-Stat, Grid-Stat, and Stat-Analysis tools are summarized using a variety of measures. MET treats probabilistic forecasts as categorical, divided into bins by user-defined thresholds between zero and one. For the categorical measures, if a forecast probability is specified in a formula, the mid-point value of the bin is used. These measures include the Brier Score (BS) with confidence bounds (Bradley 2008); the joint distribution, calibration-refinement, likelihood-base rate (Wilks 2011); and receiver operating characteristic information. Using these statistics, reliability and discrimination diagrams can be produced.

The verification statistics for probabilistic forecasts of dichotomous variables are formulated using a contingency table such as the one shown in Table C-2. In this table f represents the forecasts and o represents the observations; the two possible forecast and observation values are represented by the values 0 and 1. The values in Table C-2 are counts of the number of occurrences of all possible combinations of forecasts and observations.

Table C.2: nx2 contingency table in terms of counts. The $\mathbf{n_{ij}}$ values in the table represent the counts in each forecast-observation category, where **i** represents the forecast and **j** represents the observations. The "." symbols in the total cells represent sums across categories.

Forecast	Observation		Total
	$\mathbf{o = 1}$ (e.g., "Yes")	$\mathbf{o = 0}$ (e.g., "No")	
$\mathbf{p_1}$ = midpoint of (0 and threshold1)	$\mathbf{n_{11}}$	$\mathbf{n_{10}}$	$\mathbf{n_{1.} = n_{11} + n_{10}}$
$\mathbf{p_2}$ = midpoint of (threshold1 and threshold2)	$\mathbf{n_{21}}$	$\mathbf{n_{20}}$	$\mathbf{n_{2.} = n_{21} + n_{20}}$
. . .	. . .	. . .	. . .
$\mathbf{p_j}$ = midpoint of (threshold**i** and 1)	$\mathbf{n}$	$\mathbf{n_{i0}}$	$\mathbf{nj = n_{j1} + n_{j0}}$
Total	$n_{.1} = \sum n_{i1}$	$n_{.0} = \sum n_{i0}$	$\mathbf{T} = \sum n_i$

Reliability

Called "RELIABILITY" in PSTD output (7.11)

A component of the Brier score. Reliability measures the average difference between forecast probability and average observed frequency. Ideally, this measure should be zero as larger numbers indicate larger differences. For example, on occasions when rain is forecast with 50% probability, it should actually rain half the time.

$$\text{Reliability} = \frac{1}{T}\sum n_i(p_i - \overline{o}_i)^2$$

Resolution

Called "RESOLUTION" in PSTD output (7.11)

A component of the Brier score that measures how well forecasts divide events into subsets with different outcomes. Larger values of resolution are best since it is desirable for event frequencies in the subsets to be different than the overall event frequency.

$$\text{Resolution} = \frac{1}{T}n_{i.}(\overline{o}_i - \overline{o})^2$$

Uncertainty

Called "UNCERTAINTY" in PSTD output (7.11)

A component of the Brier score. For probabilistic forecasts, uncertainty is a function only of the frequency of the event. It does not depend on the forecasts, thus there is no ideal or better value. Note that uncertainty is equivalent to the variance of the event occurrence.

$$\text{Uncertainty} = \frac{n_{.1}}{T}(1 - \frac{n_{.1}}{T})$$

Brier score

Called "BRIER" in PSTD output (7.11)

The Brier score is the mean squared probability error. In MET, the Brier Score (BS) is calculated from the **nx2** contingency table via the following equation:

$$BS = \frac{1}{T}\sum_{i=1}^{K}\left[n_{i1}(1 - p_i)^2 + n_{i0}\,p_i^2\right]$$

The equation you will most often see in references uses the individual probability forecasts (ρ_i) and the corresponding observations ($\mathbf{o}_i$), and is given as $BS = \frac{1}{T}\sum(p_i - o_i)^2$. This equation is equivalent when the midpoints of the binned probability values are used as the $\mathbf{p_i}$.

BS can be partitioned into three terms: (1) reliability, (2) resolution, and (3) uncertainty (Murphy, 1973).

$$BS = \frac{1}{T}\sum_i (p_i - o_i)^2 = \frac{1}{T}\sum n_{i.}(p_i - \overline{o}_i)^2 - \frac{1}{T}\sum n_{i.}(\overline{o}_i - \overline{o})^2 + \overline{o}(1 - \overline{o})$$

This score is sensitive to the base rate or climatological frequency of the event. Forecasts of rare events can have a good BS without having any actual skill. Since Brier score is a measure of error, smaller values are better.

Brier Skill Score (BSS)

Called "BSS" and "BSS_SMPL" in PSTD output (7.11)

BSS is a skill score based on the Brier Scores of the forecast and a reference forecast, such as climatology. BSS is defined as

$$\text{BSS} = 1 - \frac{\text{BS}_{fcst}}{\text{BS}_{ref}}.$$

BSS is computed using the climatology specified in the configuration file while BSS_SMPL is computed using the sample climatology of the current set of observations.

OY_TP - Observed Yes Total Proportion

Called "OY_TP" in PJC output (7.12)

This is the cell probability for row **i**, column **j=1** (observed event), a part of the joint distribution (Wilks, 2011). Along with ON_TP, this set of measures provides information about the joint distribution of forecasts and events. There are no ideal or better values.

$$OYTP(i) = \frac{n_{i1}}{T} = \text{probability}(o_{i1})$$

ON_TP - Observed No Total Proportion

Called "ON_TP" in PJC output (7.12)

This is the cell probability for row **i**, column **j=0** (observed non-event), a part of the joint distribution (Wilks, 2011). Along with OY_TP, this set of measures provides information about the joint distribution of forecasts and events. There are no ideal or better values.

$$ONTP(i) = \frac{n_{i0}}{T} = \text{probability}(o_{i0})$$

Calibration

Called "CALIBRATION" in PJC output (7.12)

Calibration is the conditional probability of an event given each probability forecast category (i.e. each row in the **nx2** contingency table). This set of measures is paired with refinement in the calibration-refinement factorization discussed in Wilks (2011). A well-calibrated forecast will have calibration values that are near the forecast probability. For example, a 50% probability of precipitation should ideally have a calibration value of 0.5. If the calibration value is higher, then the probability has been underestimated, and vice versa.

$$\text{Calibration}(i) = \frac{n_{i1}}{n_{1.}} = \text{probability}(o_1|p_i)$$

Refinement

Called "REFINEMENT" in PJC output (7.12)

The relative frequency associated with each forecast probability, sometimes called the marginal distribution or row probability. This measure ignores the event outcome, and simply provides information about the frequency of forecasts for each probability category. This set of measures is paired with the calibration measures in the calibration-refinement factorization discussed by Wilks (2011).

$$\text{Refinement}(i) = \frac{n_{i.}}{T} = \text{probability}(p_i)$$

Likelihood

Called "LIKELIHOOD" in PJC output (7.12)

Likelihood is the conditional probability for each forecast category (row) given an event and a component of the likelihood-base rate factorization; see Wilks (2011) for details. This set of measures considers the distribution of forecasts for only the cases when events occur. Thus, as the forecast probability increases, so should the likelihood. For example, 10% probability of precipitation forecasts should have a much smaller likelihood value than 90% probability of precipitation forecasts.

$$\text{Likelihood}(i) = \frac{n_{i1}}{n_{.1}} = \text{probability}(p_i|o_1)$$

Likelihood values are also used to create "discrimination" plots that compare the distribution of forecast values for events to the distribution of forecast values for non-events. These plots show how well the forecasts categorize events and non-events. The distribution of forecast values for non-events can be derived from the POFD values computed by MET for the user-specified thresholds.

Base Rate

Called "BASER" in PJC output (7.12)

This is the probability of an event for each forecast category $\mathbf{p_i}$ (row), i.e. the conditional base rate. This set of measures if paired with likelihood in the likelihood-base rate factorization, see Wilks (2011) for further information. This measure is calculated for each row of the contingency table. Ideally, the event should become more frequent as the probability forecast increases.

$$\text{Base Rate}(i) = \frac{n_{i1}}{n_{i.}} = \text{probability}(o_{i1})$$

Reliability diagram

The reliability diagram is a plot of the observed frequency of events versus the forecast probability of those events, with the range of forecast probabilities divided into categories.

The ideal forecast (i.e., one with perfect reliability) has conditional observed probabilities that are equivalent to the forecast probability, on average. On a reliability plot, this equivalence is represented by the one-to-one line (the solid line in the figure below). So, better forecasts are closer to the diagonal line and worse ones are farther away. The distance of each point from the diagonal gives the conditional bias. Points that lie below the diagonal line indicate over-forecasting; in other words, the forecast probabilities are too large. The forecast probabilities are too low when the points lie above the line. The reliability diagram is conditioned on the forecasts so it is often used in combination with the ROC, which is conditioned on the observations, to provide a "complete" representation of the performance of probabilistic forecasts.

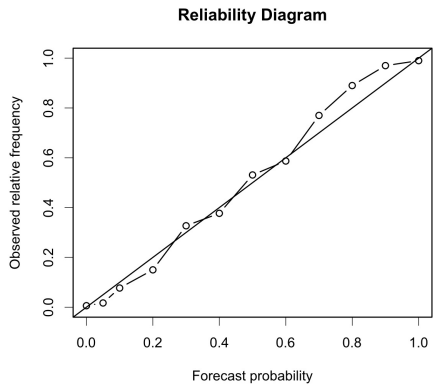

Figure C.1: Example of Reliability Diagram

Receiver operating characteristic

MET produces hit rate (POD) and false alarm rate (POFD) values for each user-specified threshold. This information can be used to create a scatter plot of POFD vs. POD. When the points are connected, the

plot is generally referred to as the receiver operating characteristic (ROC) curve (also called the "relative operating characteristic" curve). See the area under the ROC curve (AUC) entry for related information.

An ROC plot is shown for an example set of forecasts, with a solid line connecting the points for six user-specified thresholds (0.25, 0.35, 0.55, 0.65, 0.75, 0.85). The diagonal dashed line indicates no skill while the dash-dot line shows the ROC for a perfect forecast.

An ROC curve shows how well the forecast discriminates between two outcomes, so it is a measure of resolution. The ROC is invariant to linear transformations of the forecast, and is thus unaffected by bias. An unbiased (i.e., well-calibrated) forecast can have the same ROC as a biased forecast, though most would agree that an unbiased forecast is "better". Since the ROC is conditioned on the observations, it is often paired with the reliability diagram, which is conditioned on the forecasts.

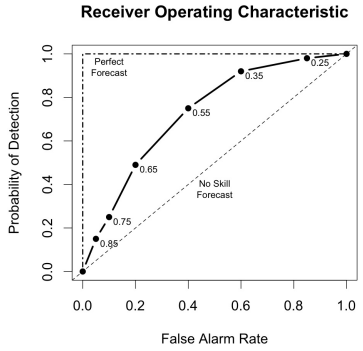

Figure C.2: Example of ROC Curve

Area Under the ROC curve (AUC)

Called "ROC_AUC" in PSTD output (7.11)

The area under the receiver operating characteristic (ROC) curve is often used as a single summary measure. A larger AUC is better. A perfect forecast has AUC=1. Though the minimum value is 0, an AUC of 0.5 indicates no skill.

The area under the curve can be estimated in a variety of ways. In MET, the simplest trapezoid method is used to calculate the area. AUC is calculated from the series of hit rate (POD) and false alarm rate (POFD) values (see the ROC entry below) for each user-specified threshold.

$$\text{AUC} = \frac{1}{2} \sum_{i=1}^{Nthresh} (POD_{i+1} + POD_i)(POFD_{i+1} - POFD_i)$$

C.4 MET verification measures for ensemble forecasts

CRPS

Called "CRPS" in ECNT output (9.2)

The continuous ranked probability score (CRPS) is the integral, over all possible thresholds, of the Brier scores (Gneiting et al, 2004). In MET, the CRPS calculation uses a normal distribution fit to the ensemble forecasts. In many cases, use of other distributions would be better.

WARNING: The normal distribution is probably a good fit for temperature and pressure, and possibly a not horrible fit for winds. However, the normal approximation will not work on most precipitation forecasts and may fail for many other atmospheric variables.

Closed form expressions for the CRPS are difficult to define when using data rather than distribution functions. However, if a normal distribution can be assumed, then the following equation gives the CRPS for each individual observation (denoted by a lowercase crps) and the corresponding distribution of forecasts.

$$crps_i(N(\mu,\sigma^2),y) = \sigma\left(\frac{y-\mu}{\sigma}\left(2\Phi\left(\frac{y-\mu}{\sigma}\right)-1\right)+2\phi\left(\frac{y-\mu}{\sigma}\right)-\frac{1}{\sqrt{\pi}}\right)$$

In this equation, the y represents the event threshold. The estimated mean and standard deviation of the ensemble forecasts (μ and σ) are used as the parameters of the normal distribution. The values of the normal distribution are represented by the probability density function (PDF) denoted by \phi and the cumulative distribution function (CDF), denoted in the above equation by ϕ.

The overall CRPS is calculated as the average of the individual measures. In equation form: CRPS = average(crps) = $\frac{1}{N}\sum_i^N crps_i$.

The score can be interpreted as a continuous version of the mean absolute error (MAE). Thus, the score is negatively oriented, so smaller is better. Further, similar to MAE, bias will inflate the CRPS. Thus, bias should also be calculated and considered when judging forecast quality using CRPS.

CRPS Skill Score

Called "CRPSS" in ECNT output (9.2)

The continuous ranked probability skill score (CRPSS) is similar to the MSESS and the BSS, in that it compares its namesake score to that of a reference forecast to produce a positively oriented score between 0 and 1.

$$\text{CRPSS} = 1 - \frac{\text{CRPS}_{fcst}}{\text{CRPS}_{ref}}$$

IGN

Called "IGN" in ECNT output (9.2)

The ignorance score (IGN) is the negative logarithm of a predictive probability density function (Gneiting et al, 2004). In MET, the IGN is calculated based on a normal approximation to the forecast distribution (i.e. a normal pdf is fit to the forecast values). This approximation may not be valid, especially for discontinuous forecasts like precipitation, and also for very skewed forecasts. For a single normal distribution **N** with parameters μ and σ, the ignorance score is

$$\text{ign}(N(\mu,\sigma),y) = \frac{1}{2}ln\big(2\pi\sigma^2\big) + \frac{(y-\mu)^2}{\sigma^2}.$$

Accumulation of the ignorance score for many forecasts is via the average of individual ignorance scores. This average ignorance score is the value output by the MET software. Like many error statistics, the IGN is negatively oriented, so smaller numbers indicate better forecasts.

PIT

Called "PIT" in ORANK output (9.7)

The probability integral transform (PIT) is the analog of the rank histogram for a probability distribution forecast (Dawid, 1984). Its interpretation is the same as that of the verification rank histogram: Calibrated probabilistic forecasts yield PIT histograms that are flat, or uniform. Under-dispersed (not enough spread in the ensemble) forecasts have U-shaped PIT histograms while over-dispersed forecasts have bell-shaped histograms. In MET, the PIT calculation uses a normal distribution fit to the ensemble forecasts. In many cases, use of other distributions would be better.

RANK

Called "RANK" in ORANK output (9.7)

The rank of an observation, compared to all members of an ensemble forecast, is a measure of dispersion of the forecasts (Hamill, 2001). When ensemble forecasts possesses the same amount of variability as the corresponding observations, then the rank of the observation will follow a discrete uniform distribution. Thus, a rank histogram will be approximately flat.

The rank histogram does not provide information about the accuracy of ensemble forecasts. Further, examination of "rank" only makes sense for ensembles of a fixed size. Thus, if ensemble members are occasionally unavailable, the rank histogram should not be used. The PIT may be used instead.

SPREAD

Called "SPREAD" in ECNT output (9.2)

Called "SPREAD" in ORANK output (9.7)

The ensemble spread for a single observation is the standard deviation of the ensemble member forecast values at that location. When verifying against point observations, these values are written to the SPREAD column of the Observation Rank (ORANK) line type. The ensemble spread for a spatial masking region is computed as the square root of the mean of the ensemble variance for all observations falling within that mask. These values are written to the SPREAD column of the Ensemble Continuous Statistics (ECNT) line type.

Note that prior to met-9.0.1, the ensemble spread of a spatial masking region was computed as the average of the spread values within that region. This algorithm was corrected in met-9.0.1 to average the ensemble variance values prior to computing the square root.

C.5 MET verification measures for neighborhood methods

The results of the neighborhood verification approaches that are included in the Grid-Stat tool are summarized using a variety of measures. These measures include the Fractions Skill Score (FSS) and the Fractions Brier Score (FBS). MET also computes traditional contingency table statistics for each combination of threshold and neighborhood window size.

The traditional contingency table statistics computed by the Grid-Stat neighborhood tool, and included in the NBRCTS output, are listed below:

* Base Rate (called "BASER" in 8.3)

* Mean Forecast (called "FMEAN" in 8.3)

* Accuracy (called "ACC" in 8.3)

* Frequency Bias (called "FBIAS" in 8.3)

* Probability of Detection (called "PODY" in 8.3)

* Probability of Detection of the non-event (called "PODN" in 8.3)

* Probability of False Detection (called "POFD" in 8.3)

* False Alarm Ratio (called "FAR" in 8.3)

* Critical Success Index (called "CSI" in 8.3)

* Gilbert Skill Score (called "GSS" in 8.3)

* Hanssen-Kuipers Discriminant (called "HK" in 8.3)

* Heidke Skill Score (called "HSS" in 8.3)

* Odds Ratio (called "ODDS" in 8.3)

All of these measures are defined in Section C.1 of Appendix C.

In addition to these standard statistics, the neighborhood analysis provides additional continuous measures, the Fractions Brier Score and the Fractions Skill Score. For reference, the Asymptotic Fractions Skill Score and Uniform Fractions Skill Score are also calculated. These measures are defined here, but are explained in much greater detail in Ebert (2008) and Roberts and Lean (2008). Roberts and Lean (2008) also present an application of the methodology.

Fractions Brier Score

Called "FBS" in NBRCNT output (8.5)

The Fractions Brier Score (FBS) is defined as FBS $= \frac{1}{N}\sum_N[\langle P_f\rangle_s - \langle P_o\rangle_s]^2$, where N is the number of neighborhoods; $\langle P_f\rangle_s$is the proportion of grid boxes within a forecast neighborhood where the prescribed threshold was exceeded (i.e., the proportion of grid boxes that have forecast events); and $\langle P_o\rangle_s$ is the proportion of grid boxes within an observed neighborhood where the prescribed threshold was exceeded (i.e., the proportion of grid boxes that have observed events).

Fractions Skill Score

Called "FSS" in NBRCNT output (8.5)

The Fractions Skill Score (FSS) is defined as

$$\text{FSS} = 1 - \frac{\text{FBS}}{\frac{1}{N}\left[\sum_N \langle P_f\rangle_s^2 + \sum_N \langle P_o\rangle_s^2\right]}$$

, where the denominator represents the worst possible forecast (i.e., with no overlap between forecast and observed events). FSS ranges between 0 and 1, with 0 representing no overlap and 1 representing complete overlap between forecast and observed events, respectively.

Asymptotic Fractions Skill Score

Called "AFSS" in NBRCNT output (8.5)

The Asymptotic Fractions Skill Score (AFSS) is a special case of the Fractions Skill score where the entire domain is used as the single neighborhood. This provides the user with information about the overall frequency bias of forecasts versus observations. The formula is the same as for FSS above, but with N=1 and the neighborhood size equal to the domain.

Uniform Fractions Skill Score

Called "UFSS" in NBRCNT output (8.5)

The Uniform Fractions Skill Score (UFSS) is a reference statistic for the Fractions Skill score based on a uniform distribution of the total forecast events across the grid. This no-skill forecast defines the UFSS, and thus a skilled forecast must have a higher value of FSS than the UFSS. Again, the formula is the same as for FSS as above, the forecast proportion in each neighborhood is the same, and is equivalent to the overall forecast event proportion.

Forecast Rate

Called "F_rate" in NBRCNT output (8.5)

The overall proportion of grid points with forecast events to total grid points in the domain. The forecast rate will match the observation rate in unbiased forecasts.

Observation Rate

Called "O_rate" in NBRCNT output (8.5)

The overall proportion of grid points with observed events to total grid points in the domain. The forecast rate will match the observation rate in unbiased forecasts. This quantity is sometimes referred to as the base rate.

C.6 MET verification measures for distance map methods

The distance map statistics include Baddeley's Δ Metric, a statistic which is a true mathematical metric. The definition of a mathematical metric is included below.

A mathematical metric, $m(A, B) \geq 0$, must have the following three properties:

1. Identity: $m(A, B) = 0$ if and only if $A = B$.
2. Symmetry: $m(A, B) = m(B, A)$

3. Triangle inequality: $m(A,C) \leq m(A,B) + m(B,C)$

The first establishes that a perfect score is zero and that the only way to obtain a perfect score is if the two sets are identical according to the metric. The second requirement ensures that the order by which the two sets are evaluated will not change the result. The third property ensures that if C is closer to A than B is to A, then $m(A,C) < M(A,B)$.

It has been argued in Gilleland (2017) that the second property of symmetry is not necessarily an important quality to have for a summary measure for verification purposes because lack of symmetry allows for information about false alarms and misses.

The results of the distance map verification approaches that are included in the Grid-Stat tool are summarized using a variety of measures. These measures include Baddeley's Δ Metric, the Hausdorff Distance, the Mean-error Distance, Pratt's Figure of Merit, and Zhu's Measure. Their equations are listed below.

Baddeley's Δ Metric and Hausdorff Distance

Called "BADDELEY" and "HAUSDORFF" in the DMAP output (8.7)

The Baddeley's Δ Metric is given by

$$\Delta_{p,w}(A,B) = [\frac{1}{N} \sum_{s \in D} |w(d(s,A)) - w(d(s,B))|]^{\frac{1}{p}}$$

where $d(s,\cdot)$is the distance map for the respective event area, $w(\cdot)$ is an optional concave function (i.e., $w(t+u) \leq w(t) + w(u)$) that is strictly increasing at zero with $w(t) = 0$ if and only if $t = 0$, N is the size of the domain, and p is a user chosen parameter for the L_p norm. The default choice of $p = 2$ corresponds to a Euclidean average, $p = 1$ is a simple average of the difference in distance maps, and the limiting case of $p = \infty$ gives the maximum difference between the two distance maps and is called the Hausdorff distance, denoted as $H(A,B)$, and is the metric that motivated the development of Baddeley's Δ metric. A typical choice, and the only available with MET, for $w(\cdot)$ is $w(t) = \min\{t, c\}$, where c is a user-chosen constant with $c = \infty$ meaning that $w(\cdot)$ is not applied. This choice of $w(\cdot)$ provides a cutoff for distances beyond the pre-specified amount given by c.

In terms of distance maps, Baddeley's Δ is the L_p norm of the top left panel in Figure 8.4 provided $c = \infty$. If $0 < c < \infty$, then the distance maps in the bottom row of Figure 8.3 would be replaced by c wherever they would otherwise exceed c before calculating their absolute differences in the top left panel of Figure 8.4.

The range for BADDELEY and HAUSDORFF is 0 to infinity, with a score of 0 indicating a perfect forecast.

Mean-error Distance

Called "MED_FO", "MED_OF", "MED_MIN", "MED_MAX", and "MED_MEAN" in the DMAP output (8.7)

The mean-error distance (MED) is given by

$$MED(A,B) = \frac{1}{n_B} \sum_{s \in B} d(x,A)$$

where n_B is the number of non-zero grid points that fall in the event set B. That is, it is the average of the distance map for the event set A calculated only over those grid points that fall inside the event set B. It gives the average shortest-distance from every point in B to the nearest point in A.

Unlike Baddeley's Δ metric, the MED is not a mathematical metric because it fails the symmetry property. However, if a metric is desired, then any of the following modifications, which are metrics, can be employed instead, and all are available through MET.

$$minMED(A,B) = min(MED(A,B), MED(B,A))$$

$$maxMED(A,B) = max(MED(A,B), MED(B,A))$$

$$meanMED(A,B) = \frac{1}{2}(MED(A,B) + MED(B,A))$$

From the distance map perspective, $MED(A,B)$ is the average of the values in Figure 8.4 (top right), and $MED(B,A)$ is the average of the values in Figure 8.4 (bottom left). Note that the average is only over the circular regions depicted in the figure.

The range for MED is 0 to infinity, with a score of 0 indicating a perfect forecast.

Pratt's Figure of Merit

Called "FOM_FO", "FOM_OF", "FOM_MIN", "FOM_MAX", and "FOM_MEAN" in the DMAP output (8.7)

Pratt's Figure of Merit (FOM) is given by

$$FOM(A,B) = \frac{1}{max(n_A, n_B)} \sum_{s \in B} \frac{1}{1 + \alpha d(s,A)^2}$$

where n_A and n_B are the number of events within event areas A and B, respectively, $d(s, A)$ is the distance map related to the event area A, and α is a user-defined scaling constant. The default, and usual choice, is $\alpha = \frac{1}{9}$ when the distances of the distance map are normalized so that the smallest nonzero distance between grid point neighbors equals one. Clearly, FOM is not a metric because like MED, it is not symmetric. Like MED, MET computes the minimum, maximum, and average of FOM_FO and FOM_OF.

Note that $d(s, A)$ in the denominator is summed only over the grid squares falling within the event set B. That is, it represents the circular area in the top right panel of Figure 8.4.

The range for FOM is 0 to 1, with a score of 1 indicating a perfect forecast.

Zhu's Measure

Called "ZHU_FO", "ZHU_OF", "ZHU_MIN", "ZHU_MAX", and "ZHU_MEAN" in the DMAP output (8.7)

Another measure incorporates the amount of actual overlap between the event sets across the fields in addition to the MED from above and was proposed by Zhu et al. (2011). Their main proposed measure was a comparative forecast performance measure of two competing forecasts against the same observation, which is not included here, but as defined is a true mathematical metric. They also proposed a similar measure of only the forecast against the observation, which is included in MET. It is simply

$$Z(A, B) = \lambda \sqrt{\frac{1}{N} \sum_{s \in D} (I_F(s) - I_O(s))^2} + (1 - \lambda) \cdot MED(A, B)$$

where $MED(A, B)$ is as in the Mean-error distance, N is the total number of grid squares as in Baddeley'sΔ metric, $I_F(s)$ ($(I_O(s))$ is the binary field derived from the forecast (observation), and λ is a user-chosen weight. The first term is just the RMSE of the binary forecast and observed fields, so it measures the average amount of overlap of event areas where zero would be a perfect score. It is not a metric because of the MED in the second term. A user might choose different weights depending on whether they want to emphasize the overlap or the MED terms more, but generally equal weight ($\lambda = \frac{1}{2}$) is sufficient. In Zhu et al (2011), they actually only consider $Z(F, O)$ and not $Z(O, F)$, but both are included in MET for the same reasons as argued with MED. Similar to MED, the average of these two directions (avg Z), as well as the min and max are also provided for convenience.

The range for ZHU is 0 to infinity, with a score of 0 indicating a perfect forecast.

C.7 Calculating Percentiles

Several of the MET tools make use of percentiles in one way or another. Percentiles can be used as part of the internal computations of a tool, or can be written out as elements of some of the standard verification statistics. There are several widely-used conventions for calculating percentiles however, so in this section we describe how percentiles are calculated in MET.

The explanation makes use of the *floor* function. The floor of a real number x, denoted $\lfloor x \rfloor$, is defined to be the greatest integer $\leq x$. For example, $\lfloor 3.01 \rfloor = 3$, $\lfloor 3.99 \rfloor = 3$, $\lfloor -3.01 \rfloor = -4$, $\lfloor -3.99 \rfloor = -4$. These examples show that the floor function does *not* simply round its argument to the nearest integer. Note also that $\lfloor x \rfloor = x$ if and only if x is an integer.

Suppose now that we have a collection of N data points x_i for $i = 0, 1, 2, \ldots, N-1$. (Note that we're using the C/C++ convention here, where array indices start at zero by default.) We will assume that the data are sorted in increasing (strictly speaking, *nondecreasing*) order, so that $i \leq j$ implies $x_i \leq x_j$. Suppose also that we wish to calculate the t percentile of the data, where $0 \leq t < 1$. For example, $t = 0.25$ for the 25th percentile of the data. Define

$$\begin{aligned} I &= \lfloor (N-1)\, t \rfloor \\ \Delta &= (N-1)\, t - I \end{aligned}$$

Then the value p of the percentile is

$$p = (1 - \Delta)\, x_I + \Delta x_{I+1}$$

Appendix D

Confidence Intervals

A single verification statistic is statistically meaningless without associated uncertainty information in accompaniment. There can be numerous sources of uncertainty associated with such a statistic including observational, physical uncertainties about the underlying processes governing the equation, sample uncertainty, etc. Although all of the sources of uncertainty can be important, the most heavily researched, and easiest to calculate, is that of sampling uncertainty. It is this source of uncertainty that can presently be obtained with MET, and the techniques for deriving these estimates are described here. Sampling uncertainty through MET is gleaned by way of confidence intervals (CIs) as these are generally most informative. A $(1-\alpha)\cdot 100\%$ confidence interval is interpreted, somewhat awkwardly, in the following way. If the test were repeated 100 times (so that we have 100 such intervals), then we expect the true value of the statistics to fall inside $(1-\alpha)\cdot 100$ of these intervals. For example, if $\alpha = 0.05$ then we expect the true value to fall within 95 of the intervals.

There are two main types of CIs available with MET: parametric and non-parametric. All of the parametric intervals used with MET rely on the underlying sample (or the errors, $F-O$) to be at least approximately independent and normally distributed. Future releases will allow for some types of dependency in the sample. The non-parametric techniques utilize what is known in the statistical literature as bootstrap resampling, which does not rely on any distributional assumptions for the sample; the assumption is that the sample is representative of the population. Bootstrap CIs can be inaccurate if the sample is not independent, but there are ways of accounting for dependence with the bootstrap, some of which will be added to MET in future releases. Details about which verification statistics have parametric CIs in MET are described next, and it should be noted that the bootstrap can be used for any statistic, though care should be taken in how it is carried out, and this is described subsequently.

The most commonly used confidence interval about an estimate for a statistic (or parameter), θ, is given by the normal approximation

$$\theta \pm z_{\alpha/2} \cdot V(\theta), \tag{D.1}$$

where $z_{\alpha/2}$ is the $\alpha-$ th quantile of the standard normal distribution, and $V(\theta)$ is the standard error of the statistic (or parameter), θ. For example, the most common example is for the mean of a sample, $X_1, \cdots, X_n$,

Table D.1: Verification statistics with normal approximation CIs given by (D.1) provided in MET along with their associated standard error estimate.

$\hat{\theta}$	$V(\theta)$
Forecast / Observation Mean	$V(\overline{X}) = \frac{\sigma_x}{\sqrt{n}}$ where σ_x emphasizes that this is the estimated standard deviation of the underlying sample.
Mean error	$V(\overline{F} - \overline{O}) = \frac{\sigma_{F-O}}{\sqrt{n}}$, where σ_{F-O} emphasizes that this is the estimated standard deviation of the errors, $F - O$.
Brier Score (BS)	$V(\text{BS}) = \frac{1}{T}\left[\Sigma F^4 + \overline{O}\left(1 - 4\Sigma F^3_{F\|O=1} + 6\Sigma F^2_{F\|O=1} - 4\Sigma F_{F\|O=1}\right) - \text{BS}^2\right]$ where **F** is the **probability** forecast and **O** is the observation. See Bradley **et al** (2008) for derivation and details.
Peirce Skill Score (PSS)	$V(\text{PSS}) = \sqrt{\frac{H(1-H)}{n_H} + \frac{F(1-F)}{n_F}}$, where **H** is the hit rate, **F** the false alarm rate, n_H the number of hits and misses, and n_F the number of false alarms and correct negatives.
Logarithm of the odds ratio (OR)	$V(\ln(OR)) = \sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}$, where the values in the denominators are the usual contingency table counts.

of independent and identically distributed (iid) normal random variables with mean μ and variance σ. Here, the mean is estimated by , and the standard error is just the standard deviation of the random variables divided by the square root of the sample size. That is, $V(\theta) = V(\overline{X}) = \frac{\sigma}{\sqrt{n}}$, and this must be estimated by $\frac{1}{n}\sum_{i=1}^{n} X_i = \overline{X}$, which is obtained here by replacing σ by its estimate, $\hat{\sigma}$, where $\hat{\sigma} = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2$.

Mostly, the normal approximation is used as an asymptotic approximation. That is, the interval (D.1) may only be appropriate for large **n**. For small **n**, the mean has an interval based on the Student's **t** distribution with **n-1** degrees of freedom. Essentially, $z_{\alpha/2}$ of (D.1) is replaced with the quantile of this **t** distribution. That is, the interval is given by

$$\mu \pm t_{\alpha/2,\nu-1} \cdot \frac{\sigma}{\sqrt{n}}, \tag{D.2}$$

where again, σ is replaced by its estimate, $\hat{\sigma}$, as described above.

Table D.1 summarizes the verification statistics in MET that have normal approximation CIs given by (D.1) along with their corresponding standard error estimates, . It should be noted that for the first two rows of this table (i.e., Forecast/Observation Mean and Mean error) MET also calculates the interval (D.2) for small sample sizes.

Other statistics in MET having parametric CIs that rely on the underlying sample to be at least approximately iid normal, but have a different form derived from the normality assumption on the sample include the variance, standard deviation, and the linear correlation coefficient. These are addressed subsequently.

Generally, the normal interval (D.1) is appropriate for statistics of continuous variables, but a limit law for the binomial distribution allows for use of this interval with proportions. The most intuitive estimate for $V(\theta)$ in this case is given by $V(p) = \sqrt{\hat{p}(1-\hat{p})/n}$. However, this only applies when the sample size is large.

A better approximation to the CI for proportions is given by Wilson's interval, which is

$$\frac{\hat{p} + z_{\alpha/2}^2 + z_{\alpha/2}\sqrt{\hat{p}(1-\hat{p})/4n}}{1 + z_{\alpha/2}^2/n}, \tag{D.3}$$

where $\hat{p}$ is the estimated proportion (e.g., hit rate, false alarm rate, PODy, PODn, etc.). Because this interval (D.3) generally works better than the more intuitive normal approximation interval for both large and small sample sizes, this is the interval employed by MET.

The forecast/observation variance has CIs derived from the underlying sample being approximately iid normal with mean μ and variance σ. The lower and upper limits for the interval are given by

$$l(\sigma^2) = \frac{(n-1)s^2}{\chi^2_{\alpha/2,n-1}} \text{ and } u(\sigma^2) = \frac{(n-1)s^2}{\chi^2_{1-\alpha/2,n-1}}, \tag{D.4}$$

respectively, where $\chi^2_{\alpha,\nu}$ is the $\alpha-$th quantile of the chi-square distribution with **n-1** degrees of freedom. Taking the square roots of the limits in (D.4) yields the CI for the forecast/observation standard deviation.

Finally, the linear correlation coefficient has limits given by

$$\left(\frac{e^{2c_l}-1}{e^{2c_l}+1}, \frac{e^{2c_u}-1}{e^{2c_u}+1}\right), \tag{D.5}$$

where $c_l = v - \frac{z_{\alpha/2}}{\sqrt{n-3}}$ and $c_u = v + \frac{z_{\alpha/2}}{\sqrt{n-3}}$.

All other verification scores with CIs in MET must be obtained through bootstrap resampling. However, it is also possible to obtain bootstrap CIs for any of the statistics given above, and indeed it has been proven that the bootstrap intervals have better accuracy for the mean than the normal approximation. The bootstrap algorithm is described below.

1. Assume the sample is representative of the population.
2. Resample with replacement from the sample (see text below).
3. Estimate the parameter(s) of interest for the current replicated sample.
4. Repeat steps 2 and 3 numerous times, say B times, so that you now have a sample of size B of the parameter(s).
5. Calculate CIs for the parameters directly from the sample (see text below for more details)

Typically, a simple random sample is taken for step 2, and that is how it is done in MET. As an example of what happens in this step, suppose our sample is X_1, X_2, X_3, X_4. Then, one possible replicate might be X_2, X_2, X_2, X_4. Usually one samples $m = n$ points in this step, but there are cases where one should use $m < n$. For example, when the underlying distribution is heavy-tailed, one should use a smaller size m than n (e.g., the closest integer value to the square root of the original sample size).

There are numerous ways to construct CIs from the sample obtained in step 4. MET allows for two of these procedures: the percentile and the BCa. The percentile is the most commonly known method, and the simplest to understand. It is merely the $\alpha/2$ and $1 - \alpha/2$ percentiles from the sample of statistics. Unfortunately, however, it has been shown that this interval is too optimistic in practice (i.e., it doesn't have accurate coverage). One solution is to use the BCa method, which is very accurate, but it is also computationally intensive. This method adjusts for bias and non-constant variance, and yields the percentile interval in the event that the sample is unbiased with constant variance.

If there is dependency in the sample, then it is prudent to account for this dependency in some way. One method that does not make a lot of assumptions is circular block bootstrapping. This is not currently implemented in MET, but will be available in a future release. At that time, the method will be explained more fully here, but until then consult Gilleland (2010) for more details.

Appendix E

WWMCA Tools

There are two WWMCA tools available. The `wwmca_plot` tool makes a PostScript plot of one or more WWMCA cloud percent files and the `wwmca_regrid` tool regrids WWMCA cloud percent files and reformats them into netCDF files that the other MET tools can read.

The WWMCA tools get valid time and hemisphere (north or south) information from the file names, so it's important for both of the WWMCA tools that these file names not be changed.

The usage statement for `wwmca_plot` is

```
wwmca_plot [ -outdir path ] wwmca_cloud_pct_file_list
```

Here, `wwmca_cloud_pct_file_list` represents one or more WWMCA cloud percent files given on the command line. As with any command given to a UNIX shell, the user can use meta-characters as a shorthand way to specify many filenames.

The optional `-outdir` argument specifies a directory where the output PostScript plots will be placed. If not specified, then the plots will be put in the current (working) directory. Figure C.2 shows an example of the `wwmca_plot` output.

The usage statement for **wwmca_regrid** is

```
wwmca_regrid -out filename config filename [ -nh filename ] [ -sh filename ]
```

Here, the **-out** switch tells **wwmca_regrid** what to name the output netCDF file. The **-config** switch gives the name of the config file that **wwmca_regrid** should use—like many of the MET tools, **wwmca-regrid** uses a configuration file to specify user-changeable parameters. The format of this file will be explained below.

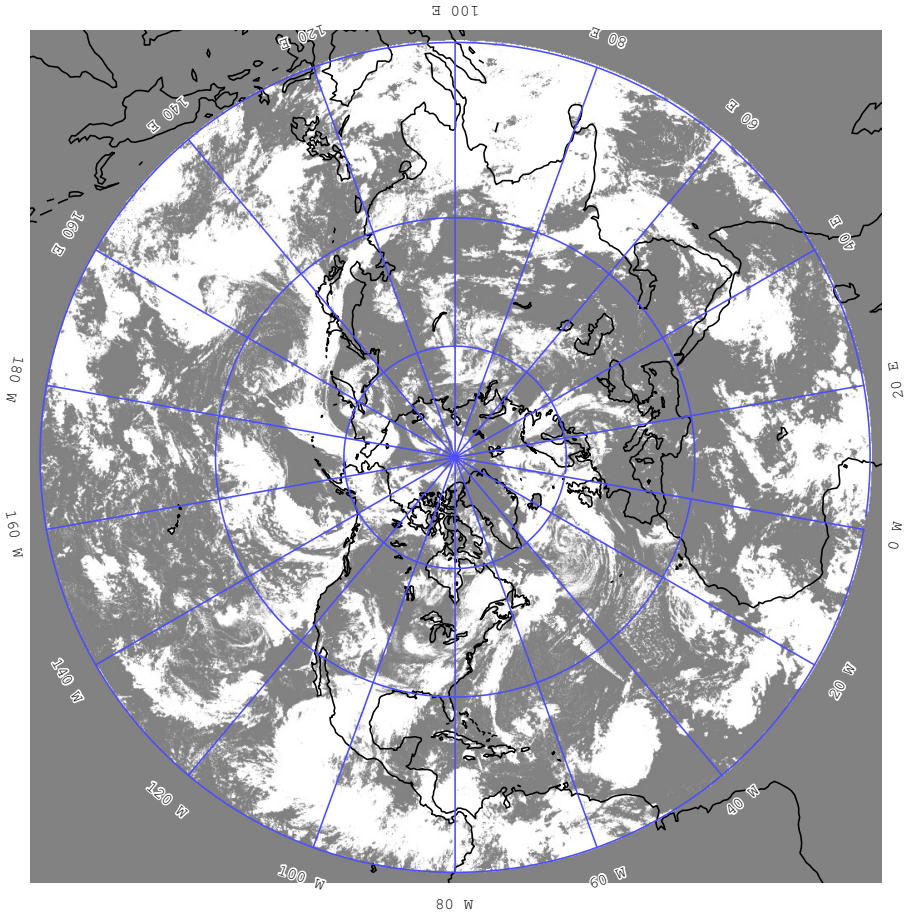

Figure C.2: Example output of `wwmca_plot` tool.

The **-nh** and **-sh** options give names of WWMCA cloud percent files that **wwmca_regrid** should use as input. Northern hemisphere files are specified with **-nh**, and southern hemisphere files with **-sh**. At least one of these must be given, but in many cases both need not be given.

In any regridding problem, there are two grids involved: the "From" grid, which is the grid the input data are on, and the "To" grid, which is the grid the data are to be moved onto. For **wwmca_regrid**, the "To" grid is specified in the config file. If this grid is entirely confined to one hemisphere, then only theWWMCA data file for that hemisphere need be given. It's only when the "To" grid straddles the equator that data files for both hemispheres need be given (though the interpolation width parameter in the config file can change this—see below). Once the "To" grid is specified in the config file, the **wwmca_regrid** tool will know which input data files it needs, and will complain if it's not given the right ones.

Now let's talk about the details of the config file. The config file has the same C-like syntax that all the other MET config files use. The first (and most complicated) thing to specify is the "To" grid. This is given by the **To_grid** parameter. If you are using one of the standard NCEP grids, for example grid #218, you can simply write

```
To grid = "G218";
```

and that will work. Failing that, you must give the parameters that specify the grid and it's projection. Four projections are supported: Lambert Conformal, Polar Stereographic, Plate Carrée, and Mercator.

To specify a Lambert Grid, the syntax is

```
lambert Nx Ny lat_ll lon_ll lon_orient D_km R_km standard_lat_1 [ standard_lat_2 ] N|S
```

Here, **Nx** and **Ny** are the number of points in, respectively, the **x** and **y** grid directions. These two numbers give the overall size of the grid. **lat_ll** and **lon_ll** are the latitude and longitude, in degrees, of the lower left point of the grid. North latitude and east longitude are considered positive. **lon_orient** is the orientation longitude of the grid. It's the meridian of longitude that's parallel to one of the vertical grid directions. **D_km** and **R_km** are the grid resolution and the radius of the Earth, both in kilometers. **standard_lat_1** and **standard_lat_2** are the standard parallels of the Lambert projection. If the two latitudes are the same, then only one need be given. **N|S** means to write either **N** or **S** depending on whether the Lambert projection is from the north pole or the south pole.

As an example of specifying a Lambert grid, suppose you have a northern hemisphere Lambert grid with 614 points in the x direction and 428 points in the y direction. The lower left corner of the grid is at latitude 12.190° north and longitude 133.459° west. The orientation longitude is 95° west. The grid spacing is 12.19058° km. The radius of the Earth is the default value used in many grib files: 6367.47 km. Both standard parallels are at 25° north. To specify this grid in the config file, you would write

```
To grid = "lambert 614 428 12.190 -133.459 -95.0 12.19058 6367.47 25.0 N";
```

For a Polar Stereographic grid, the syntax is

```
Nx Ny lat_ll lon_ll lon_orient D_km R_km lat_scale N|S
```

Here, **Nx, Ny, lat_ll, lon_ll, lon_orient, D_km** and **R_km** have the same meaning as in the Lambert case. **lat_scale** is the latitude where the grid scale **D_km** is true, while **N|S** means to write either **N** or **S** depending on whether the stereographic projection is from the north pole or the south pole.

For Plate Carrée grids, the syntax is

```
latlon Nx Ny lat_ll lon_ll delta_lat delta_lon
```

The parameters **Nx, Ny, lat_ll** and **lon_ll** are as before. **delta_lat** and **delta_lon** are the latitude and longitude increments of the grid—i.e., the change in latitude or longitude between one grid point and an adjacent grid point.

For a Mercator grid, the syntax is

```
mercator Nx Ny lat_ll lon_ll lat_ur lon_ur
```

The parameters **Nx, Ny, lat_ll** and **lon_ll** are again as before, while **lat_ur** and **lon_ur** are the latitude and longitude of the upper right corner of the grid.

Thankfully, the rest of the parameters in the config file are easier to specify.

The next two config file parameters have to do with specifying the interpolation scheme used. The **interp_method** parameter specifies which interpolation method is to be used. Four methods are supported: average, maximum, minimum and nearest neighbor. As an example, to specify the "average" method, one would write

```
interp_method = "average";
```

The other interpolation parameter is **interp_width**. This specifies the width of the interpolation box used in the above interpolation method. An example value could be

```
interp_width = 5;
```

The value must be odd and ≥ 1. If a value of 1 is specified, then nearest neighbor interpolation will be used regardless of the value assigned to **interp_method**.

The fact that an interpolation box is used has one subtle implication—the "To" grid is effectively fattened by half the width of the interpolation box. This means that even for a "To" grid that is entirely contained in one hemisphere, if it comes close to the equator, this virtual fattening may be enough to push it over the equator, and the user will then have to provide inputWWMCA files for both hemispheres, even though the "To" grid doesn't cross the equator. The **wwmca_regrid** tool should detect this situation and complain to the user if not given the correct input files.

The next variable, **good_percent**, tells what fraction of the values in the interpolation square need to be "good" in order for the interpolation scheme to return a "good" result. Example:

```
good percent = 0;
```

The rest of the config file parameters have to do with how the output netCDF file represents the data. These should be self-explanatory, so I'll just give an example:

```
variable_name = "Cloud Pct";
long_name     = "cloud cover percent";
grib_code     = 100;
units         = "percent";
level         = "SFC";
```

Appendix F

Python Embedding

F.1 Introduction

MET includes the ability to embed Python to a limited degree. Users may use Python scripts and whatever associated Python packages they wish in order to prepare 2D gridded data fields, point observations, and matched pairs as input to the MET tools. We fully expect that this degree of embedding will increase in the future. In addition, plans are in place to extend Python with MET in upcoming releases, allowing users to invoke MET tools directly from their Python script. While MET version 8.0 was built on Python 2.x, MET version 9.0 is build on Python 3.6+.

F.2 Compiling Python Support

In order to use Python embedding, the user's local Python installation must have the C-language Python header files and libraries. Sometimes when Python is installed locally, these header files and libraries are deleted at the end of the installation process, leaving only the binary executable and run-time shared object files. But the Python header files and libraries must be present to compile support in MET for Python embedding. Assuming the requisite Python files are present, and that Python embedding is enabled when building MET (which is done by passing the `--enable-python` option to the `configure` command line), the MET C++ code will use these in the compilation process to link directly to the Python libraries.

In addition to the `configure` option mentioned above, two variables, `MET_PYTHON_CC` and `MET_PYTHON_LD`, must also be set for the configuration process. These may either be set as environment variables or as command line options to `configure`. These constants as passed as compiler command line options when building MET to enable the compiler to find the requisite Python header files and libraries in the user's local filesystem. Fortunately, Python provides a way to set these variables properly. This frees the user from the necessity of having any expert knowledge of the compiling and linking process. Along with the `Python` executable, there should be another executable called `python-config,` whose output can be used to set these environment variables as follows:

- On the command line, run "`python-config --cflags`". Set the value of `MET_PYTHON_CC` to the output of that command.
- Again on the command line, run "`python-config --ldflags`". Set the value of `MET_PYTHON_LD` to the output of that command.

Make sure that these are set as environment variables or that you have included them on the command line prior to running `configure`.

F.3 MET_PYTHON_EXE

When Python embedding support is compiled, MET instantiates the Python interpreter directly. However, for users of highly configurable Conda environments, the Python instance set at compilation time may not be sufficient. Users may want to switch between Conda environments for which different packages are available. MET version 9.0 has been enhanced to address this need.

The types of Python embedding supported in MET are described below. In all cases, by default, the compiled Python instance is used to execute the Python script. If the packages that script imports are not available for the compiled Python instance, users will encounter a runtime error. In the event of a runtime error, users are advised to set the **MET_PYTHON_EXE** environment variable and rerun. This environment variable should be set to the full path to the version of Python you would like to use. See an example below.

```
export MET_PYTHON_EXE=/usr/local/python3/bin/python3
```

Setting this environment variable triggers slightly different processing logic in MET. Rather than executing the user-specified script with compiled Python instance directly, MET does the following:

1. Wrap the user's Python script and arguments with a wrapper script (write_pickle_mpr.py, write_pickle_point.py, or write_pickle_dataplane.py) and specify the name of a temporary file to be written.
2. Use a system call to the MET_PYTHON_EXE Python instance to execute these commands and write the resulting data objects to a temporary Python pickle file.
3. Use the compiled Python instance to read data from that temporary pickle file.

With this approach, users should be able to execute Python scripts in their own custom environments.

F.4 Python Embedding for 2D data

We now describe how to write Python scripts so that the MET tools may extract 2D gridded data fields from them. Currently, MET offers two ways to interact with Python scripts: by using NumPy arrays or by using Xarray objects. The interface to be used (NumPy or Xarray) is specified on the command line (more on this later). The user's scripts can use any Python libraries that are supported by the local Python installation, or any personal or institutional libraries or code that are desired in order to implement the Python script, so long as at the data has been loaded into either a NumPy array or an Xarray object by the end of the script. This offers advantages when using data file formats that MET does not directly support. If there is Python code to read the data format, the user can use those tools to read the data, and then copy the data into a NumPy array or an Xarray object. MET can then ingest the data via the Python script. Note that whether a NumPy array or an Xarray object is used, the data should be stored as double precision floating point numbers. Using different data types, such as integers or single precision floating point numbers, will lead to unexpected results in MET.

Using NumPy. The data must be loaded into a 2D NumPy array named `met_data`. In addition there must be a Python dictionary named `attrs` which contains metadata such as timestamps, grid projection and other information. Here is an example `attrs` dictionary:

```
attrs = {

   'valid':      '20050807_120000',
   'init':       '20050807_000000',
   'lead':       '120000',
   'accum':      '120000',

   'name':       'Foo',
   'long_name':  'FooBar',
   'level':      'Surface',
   'units':      'None',

   'grid': {
      'type': 'Lambert Conformal',
      'hemisphere': 'N',
      'name': 'FooGrid',
      'scale_lat_1': 25.0,
      'scale_lat_2': 25.0,
      'lat_pin': 12.19,
      'lon_pin': -135.459,
      'x_pin': 0.0,
      'y_pin': 0.0,
      'lon_orient': -95.0,
      'd_km': 40.635,
      'r_km': 6371.2,
```

```
      'nx': 185,
      'ny': 129,
   }

}
```

In the dictionary, valid time, initialization time, lead time and accumulation time (if any) must be indicated by strings. Valid and initialization times must be given in YYYYMMDD[_HH[MMSS]] format, and lead and accumulation times must be given in HH[MMSS] format, where the square brackets indicate optional elements. The dictionary must also include strings for the name, long_name, level, and units to describe the data. The rest of the `attrs` dictionary gives the grid size and projection information in the same format that is used in the netCDF files written out by the MET tools. Note that the `grid` entry in the `attrs` dictionary is itself a dictionary.

Using Xarray Objects. To use Xarray objects, a similar procedure to the NumPy case is followed. An Xarray object has a NumpyArray called `values`, and an attributes dictionary called `attrs`. The user must name the Xarray object to be `met_data`. When one of the MET tools runs the Python script, it will look for an Xarray object named `met_data`, and will retrieve the data and metadata from the `values` and `attrs` parts, respectively, of the Xarray object. The Xarray `attrs` dictionary is populated in the same way as for the NumPy interface. The `values` Numpy array part of the Xarray object is also populated in the same way as the NumPy case.

It remains to discuss command lines and config files. Two methods for specifying the Python command and input file name are supported.

Python Embedding Option 1:

On the command line for any of the MET tools which will be obtaining its data from a Python script rather than directly from a data file, the user should specify either PYTHON_NUMPY or PYTHON_XARRAY wherever a (forecast or observation) data file name would normally be given. Then in the `name` entry of the config file dictionaries for the forecast or observation data, the user should list the Python script to be run followed by any command line arguments for that script. Note that for tools like MODE that take two data files, it would be entirely possible to use the NumPy interface for one file and the Xarray interface for the other.

Listed below is an example of running the `plot_data_plane` tool to call a Python script for data that is included with the MET release tarball. Assuming the MET executables are in your path, this example may be run from the top-level MET source code directory.

```
plot_data_plane PYTHON_NUMPY fcst.ps \
  'name="scripts/python/read_ascii_numpy.py data/python/fcst.txt FCST";' \
  -title "Python enabled plot_data_plane"
```

The first argument for the `plot_data_plane` tool is the gridded data file to be read. When calling a NumPy Python script, set this to the constant string PYTHON_NUMPY. The second argument is the name of the output PostScript file to be written. The third argument is a string describing the data to be plotted. When calling a Python script, set `name` to the Python script to be run along with command line arguments. Lastly, the `-title` option is used to add a title to the plot. Note that any print statements included in the Python script will be printed to the screen. The above example results in the following log messages.

```
DEBUG 1: Opening data file: PYTHON_NUMPY
Input File: 'data/python/fcst.txt'
Data Name : 'FCST'
Data Shape: (129, 185)
Data Type:  dtype('float64')
Attributes: {'name': 'FCST',  'long_name': 'FCST_word',
             'level': 'Surface', 'units': 'None',
             'init': '20050807_000000', 'valid': '20050807_120000',
             'lead': '120000',  'accum': '120000'
             'grid': {...} }
DEBUG 1: Creating postscript file: fcst.ps
```

Python Embedding Option 2 using MET_PYTHON_INPUT_ARG:

The second option was added to support the use of Python embedding in tools which read multiple input files. Option 1 reads a single field of data from a single source, whereas tools like Ensemble-Stat, Series-Analysis, and MTD read data from multiple input files. While option 2 can be used in any of the MET tools, it is required for Python embedding in Ensemble-Stat, Series-Analysis, and MTD.

On the command line for any of the MET tools, specify the path to the input gridded data file(s) as the usage statement for the tool indicates. Do **not** substitute in PYTHON_NUMPY or PYTHON_XARRAY on the command line. In the config file dictionary set the `file_type` entry to either PYTHON_NUMPY or PYTHON_XARRAY to activate the Python embedding logic. Then, in the `name` entry of the config file dictionaries for the forecast or observation data, list the Python script to be run followed by any command line arguments for that script. However, in the Python command, replace the name of the input gridded data file with the constant string **MET_PYTHON_INPUT_ARG**. When looping over multiple input files, the MET tools will replace that constant MET_PYTHON_INPUT_ARG with the path to the file currently being processed. The example `plot_data_plane` command listed below yields the same result as the example shown above, but using the option 2 logic instead.

```
plot_data_plane data/python/fcst.txt fcst.ps \
  'name="scripts/python/read_ascii_numpy.py MET_PYTHON_INPUT_ARG FCST"; \
   file_type=PYTHON_NUMPY;' \
  -title "Python enabled plot_data_plane"
```

F.5 Python Embedding for Point Observations

The ASCII2NC tool supports the "-format python" option. With this option, point observations may be passed as input. An example of this is provided in Section 4.2.1.1. That example uses the **read_ascii_point.py** sample script which is included with the MET code. It reads ASCII data in MET's 11-column point observation format and stores it in a Pandas dataframe to be read by the ASCII2NC tool with Python.

The **read_ascii_point.py** sample script can be found in:

- MET installation directory in **MET_BASE/python**.
- MET GitHub repository (`https://github.com/NCAR/MET`) in **met/scripts/python**.

F.6 Python Embedding for MPR data

The Stat-Analysis tool supports the "-lookin python" option. With this option, matched pair (MPR) data may be passed as input. An example of this is provided in Section 12.3.1.1. That example uses the **read_ascii_mpr.py** sample script which is included with the MET code. It reads MPR data and stores it in a Pandas dataframe to be read by the Stat-Analysis tool with Python.

The **read_ascii_mpr.py** sample script can be found in:

- MET installation directory in **MET_BASE/python**.
- MET GitHub repository (`https://github.com/NCAR/MET`) in **met/scripts/python**.

Appendix G

Vectors and Vector Statistics

In this appendix, we discuss some basic properties of vectors, concentrating on the two-dimensional case. To keep the discussion simple, we will assume we are using a Cartesian coordinate system.

Traditionally, vectors have been defined as quantities having both magnitude and direction, exemplified by a directed line segment. The magnitude of the vector is shown by the length of the segment, and the direction of the vector is usually shown by drawing an arrowhead on one end of the segment. Computers (and, in the author's experience, people) are usually more comfortable working with numbers, and so instead of the usual graphical definition of a vector, we will take the definition used in analytic geometry: A (two-dimensional) vector is an ordered pair of numbers. We will use boldface type to denote vectors, and so we can write

$$\mathbf{v} = (a, b)$$

to show that the vector $\mathbf{v}$ consists of the ordered pair of numbers a and b. The number a is called the first (or x) component of $\mathbf{v}$, and b is called the second (or y) component. Vector addition is performed component-wise: $(a, b) + (c, d) = (a + c, b + d)$, and similarly for subtraction. If α is a scalar, then we define multiplication by the scalar α as $\alpha(a, b) = (\alpha a, \alpha b)$, and similarly for division by a (nonzero!) scalar.

The *norm* (or length, or magnitude) of a vector $\mathbf{v} = (a, b)$, is

$$|\mathbf{v}| = \sqrt{a^2 + b^2}$$

Note that $|\mathbf{v}| = 0$ if and only if $a = b = 0$, in which case we say that $\mathbf{v}$ is the *zero vector*. If α is a scalar, then

$$|\alpha\mathbf{v}| = |\alpha|\,|\mathbf{v}|$$

The most important relation between vectors and their norms is given by the *triangle inequality*:

$$|\mathbf{v} + \mathbf{w}| \leq |\mathbf{v}| + |\mathbf{w}|$$

In some cases, only the direction of a vector is of interest, and in such cases we can replace a nonzero vector $\mathbf{v}$ by the unique vector $N(\mathbf{v})$ that has the same direction as $\mathbf{v}$, but has norm 1:

$$N(\mathbf{v}) = \frac{\mathbf{v}}{|\mathbf{v}|}$$

The vector $N(\mathbf{v})$ will be called the *unit vector* corresponding to $\mathbf{v}$, or more simply the *direction* of $\mathbf{v}$. Note that the zero vector has no direction.

Since vectors are characterized by magnitude (norm) and direction, this gives two ways to compare vectors: we can compare either their magnitudes or their directions. If $\mathbf{v}$ and $\mathbf{w}$ are vectors, then we can compare their norms by either taking the norm of the difference $|\mathbf{v} - \mathbf{w}|$ or the difference of the norms $|\mathbf{v}| - |\mathbf{w}|$. It's not always made clear in verification studies which of these is meant, and in general these two quantities will be different. However, by making use of the triangle inequality it can be shown that there is a relation between them. To derive this, let $\mathbf{z} = \mathbf{v} - \mathbf{w}$, from which we get $\mathbf{v} = \mathbf{w} + \mathbf{z}$. Now taking norms and using the triangle inequality,

$$|\mathbf{v}| = |\mathbf{w} + \mathbf{z}| \leq |\mathbf{w}| + |\mathbf{z}| = |\mathbf{w}| + |\mathbf{v} - \mathbf{w}|$$

which gives

$$|\mathbf{v}| - |\mathbf{w}| \leq |\mathbf{v} - \mathbf{w}|$$

Reversing the roles of $\mathbf{v}$ and $\mathbf{w}$ now gives the result:

$$\big|\, |\mathbf{v}| - |\mathbf{w}| \,\big| \leq |\mathbf{v} - \mathbf{w}|$$

In the same manner, we can compare the directions of two different nonzero vectors $\mathbf{v}$ and $\mathbf{w}$ by either the direction of the difference $N(\mathbf{v} - \mathbf{w})$, or by the difference in the directions $N(\mathbf{v}) - N(\mathbf{w})$. Unlike the case for magnitudes, however, there is in general no relationship at all between these two measures of direction difference.

Now let us specialize this discussion of vectors to verification of wind vector data. We will denote the forecast wind vector by $\mathbf{F}$, and the observed wind vector by $\mathbf{O}$. These are two-dimensional horizontal vectors with u and v components as follows:

$$\mathbf{F} = (u_f, v_f) \qquad \text{and} \qquad \mathbf{O} = (u_o, v_o)$$

We will assume that we have N observations of forecast and observed wind:

$$\mathbf{F}_i = (u_{fi}, v_{fi}) \qquad \text{and} \qquad \mathbf{O}_i = (u_{oi}, v_{oi})$$

for $1 \leq i \leq N$. We also have the forecast and observed wind *speeds:*

$$s_f = |\mathbf{F}| = \sqrt{u_f^2 + v_f^2} \qquad \text{and} \qquad s_o = |\mathbf{O}| = \sqrt{u_o^2 + v_o^2}$$

and, at each data point,

$$s_{fi} = |\mathbf{F}_i| = \sqrt{u_{fi}^2 + v_{fi}^2} \qquad \text{and} \qquad s_{oi} = |\mathbf{O}_i| = \sqrt{u_{oi}^2 + v_{oi}^2}$$

It will be convenient to denote the average forecast and observed wind vectors by $\mathbf{F}_a$ and $\mathbf{O}_a$:

$$\mathbf{F}_a = \frac{1}{N}\sum_i \mathbf{F}_i \qquad \text{and} \qquad \mathbf{O}_a = \frac{1}{N}\sum_i \mathbf{O}_i$$

Now let us look at the definitions of the vector statistics produced by MET:

FBAR and OBAR are the average values of the forecast and observed wind speed.

$$\text{FBAR} = \frac{1}{N}\sum_i s_{fi}$$

$$\text{OBAR} = \frac{1}{N}\sum_i s_{oi}$$

FS_RMS and OS_RMS are the root-mean-square values of the forecast and observed wind speeds.

$$\text{FS\_RMS} = \left[\frac{1}{N}\sum_i s_{fi}^2\right]^{1/2}$$

$$\text{OS\_RMS} = \left[\frac{1}{N}\sum_i s_{oi}^2\right]^{1/2}$$

MSVE and RMSVE are, respectively, the mean squared, and root mean squared, lengths of the vector difference between the forecast and observed wind vectors.

$$\text{MSVE} = \frac{1}{N}\sum_i |\mathbf{F}_i - \mathbf{O}_i|^2$$

$$\text{RMSVE} = \sqrt{\text{MSVE}}$$

FSTDEV and OSTDEV are the standard deviations of the forecast and observed wind speeds. In these equations, μ_f and μ_o are the average forecast and observed wind speeds

$$\text{FSTDEV} = \frac{1}{N}\sum_i (s_{fi} - \text{FBAR})^2 = \frac{1}{N}\sum_i s_{fi}^2 - \text{FBAR}^2$$

$$\text{OSTDEV} = \frac{1}{N}\sum_i (s_{oi} - \text{OBAR})^2 = \frac{1}{N}\sum_i s_{oi}^2 - \text{OBAR}^2$$

FDIR and ODIR are the direction (angle) of $\mathbf{F}_a$ and $\mathbf{O}_a$ with respect to the grid directions.

$$\text{FDIR} = \text{direction angle of } \mathbf{F}_a$$

$$\text{ODIR} = \text{direction angle of } \mathbf{O}_a$$

FBAR_SPEED and OBAR_SPEED are the lengths of the average forecast and observed wind vectors. Note that this is *not* the same as the average forecast and observed wind speeds (*ie.,* the length of an average vector $\neq$ the average length of the vector).

$$\text{FBAR\_SPEED} = |\mathbf{F}_a|$$

$$\text{OBAR\_SPEED} = |\mathbf{O}_a|$$

VDIFF_SPEED is the length (*ie. speed* of the vector difference between the average forecast and average observed wind vectors.

$$\text{VDIFF\_SPEED} = |\mathbf{F}_a - \mathbf{O}_a|$$

Note that this is *not* the same as the difference in lengths (speeds) of the average forecast and observed wind vectors. That quantity is called SPEED_ERR (see below). There is a relationship between these two statistics however: using some of the results obtained in the introduction to this appendix, we can say that $\big|\,|\mathbf{F}_a| - |\mathbf{O}_a|\,\big| \leq |\mathbf{F}_a - \mathbf{O}_a|$ or , equivalently, that $|\text{SPEED\_ERR}| \leq \text{VDIFF\_SPEED}$.

VDIFF_DIR is the direction of the vector difference of the average forecast and average observed wind vectors. Note that this is *not* the same as the difference in direction of the average forecast and average observed wind vectors. This latter quantity would be FDIR – ODIR.

$$\text{VDIFF\_DIR} = \text{direction of } (\mathbf{F}_a - \mathbf{O}_a)$$

SPEED_ERR is the difference in the lengths (speeds) of the average forecast and average observed wind vectors. (See the discussion of VDIFF_SPEED above.)

$$\text{SPEED\_ERR} = |\mathbf{F}_a| - |\mathbf{O}_a| = \text{FBAR\_SPEED} - \text{OBAR\_SPEED}$$

SPEED_ABSERR is the absolute value of SPEED_ERR. Note that we have SPEED_ABSERR $\leq$ VDIFF_SPEED (see the discussion of VDIFF_SPEED above).

$$\text{SPEED\_ABSERR} = |\text{SPEED\_ERR}|$$

DIR_ERR is the signed angle between the directions of the average forecast and average observed wind vectors. Positive if the forecast vector is counterclockwise from the observed vector.

$$\text{DIR\_ERR} = \text{direction between } N(\mathbf{F}_a) \text{ and } N(\mathbf{O}_a)$$

DIR_ABSERR is the absolute value of DIR_ERR. In other words, it's an unsigned angle rather than a signed angle.

$$\text{DIR\_ABSERR} = |\text{DIR\_ERR}|$$
