# Model Evaluation Tools Version 0.9 (METv0.9)

# User's Guide

**Developmental Testbed Center**
**Research Applications Laboratory**
**National Center for Atmospheric Research**

**Boulder, Colorado, USA**

**July 2007**

## Contents

## Foreword: A note to MET users

This user's guide is provided as an aid to users of the Model Evaluation Tools (MET). The MET is a set of verification tools developed by the Developmental Testbed Center (DTC) at the National Center for Atmospheric Research (NCAR) for the numerical weather prediction community – and especially users and developers of the Weather Research and Forecasting (WRF) model – 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. The beta release of MET (METv0.9) was on 16 July 2007. Future releases are expected in Fall 2007 and January 2008, and will include new capabilities and enhancements as well as corrections to any errors or system issues.  In addition, MET will be able to accept new modules contributed by the community. A protocol will be established to determine the maturity of new verification methods that are contributed and to coordinate the inclusion of new modules in future versions.

This user's guide was prepared by the developers of the MET, including John Halley Gotway, Lacey Holland, Barbara Brown, Randy Bullock, David Ahijevych, and Eric Gilleland.

## Acknowledgments

We thank the U.S. Air Force Weather Agency for their support of this work. Thanks also go to the staff at the Developmental Testbed Center for their help, advice, and many types of support. Many individuals participated in Town Hall meetings, at the 2007 Annual Meeting of the American Meteorological Society and at NCAR, and gave us valuable insight into the needs of the WRF user community; we appreciate their effort to participate and are thankful for their advice and ideas. We are also very grateful to the individuals who participated in a MET planning workshop in February 2007; the ideas generated at that workshop will help MET grow in future years into a truly useful and state-of-the-art system. Finally, we would like to specifically thank Elizabeth Ebert and Tressa Fowler for their guidance and other contributions. Tressa provided early leadership for this project and organized and coordinated the town hall meetings and workshop. Beth Ebert generously helped to organize the workshop and provided an immense amount of helpful advice along the way. NCAR is sponsored by the National Science Foundation.

# 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 users to apply MET to their specific datasets and evaluation studies. The MET has been specifically designed for application to the Weather Research and Forecast (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 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 – 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 focuses on the data needed to run MET, including formats for forecasts, observations, and output. Chapters 4 through 6 focus on the main modules contained in the current version of MET, including the Point-Stat, Grid-Stat, and MODE 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 form of the output. Finally, Chapters 7 through 9 include some additional tools and information for scripting, analysis, and plotting. Version 0.9 of MET has limited capabilities for analysis and plotting; these capabilities will be enhanced in future versions. 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). In the future, Appendix C will describe and link to code that can be used to create particular types of plots of verification results (note that the MET development group will accept contributed code which will be described in this appendix).

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/) at the National Center for Atmospheric Research (NCAR) in Boulder, Colorado. 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 a functionally equivalent operational environment for the research community to test model enhancements, and the operational community benefits from DTC testing of model and evaluation before new models are implemented operationally. MET will also serve 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 DTC to evaluate the strengths and weaknesses of advances in NWP prior to operational implementation.

For example, one of the DTC's charges is to maintain the WRF Reference Code. Unlike WRF Contributed Code, which meets minimum programming standards and has been tested by its contributors, WRF Reference Code consists of NWP code that has been rigorously tested by the DTC and meets specific quality standards. When Contributed Code is considered for elevation to Reference Code status, the DTC must assess the proposed changes by analyzing effects on forecast performance. MET will aid in the analysis of WRF Reference Code changes through application of multiple methods of evaluating NWP forecasts.

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 the MET will incorporate newly developed and advanced methodologies, including new methods for diagnostic and spatial verification and new techniques provided by the verification and modeling communities. The MET also utilizes and replicates the capabilities of existing systems for verification of NWP forecasts. For example, the MET package will replicate existing 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 an initial capability for spatial forecast verification. The MET will include new state-of-the-art 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 – from the NWP and verification communities as well as the DTC's MET developers – 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 might 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.

The MET code and documentation will be maintained by the DTC in Boulder, Colorado. The MET package is freely available to the modeling, verification, and operational communities, including universities, 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, intermediate output, statistical analyses, and output. Each of these stages is considered further in later chapters. For example, the input and output formats are discussed in Chapter 2 as well as in the chapters associated with each of the statistics modules. MET input files are represented on the far left. Note that forecast model output is currently expected to be in GRIB1 format; GRIB2 and other formats will be incorporated in future releases of MET.

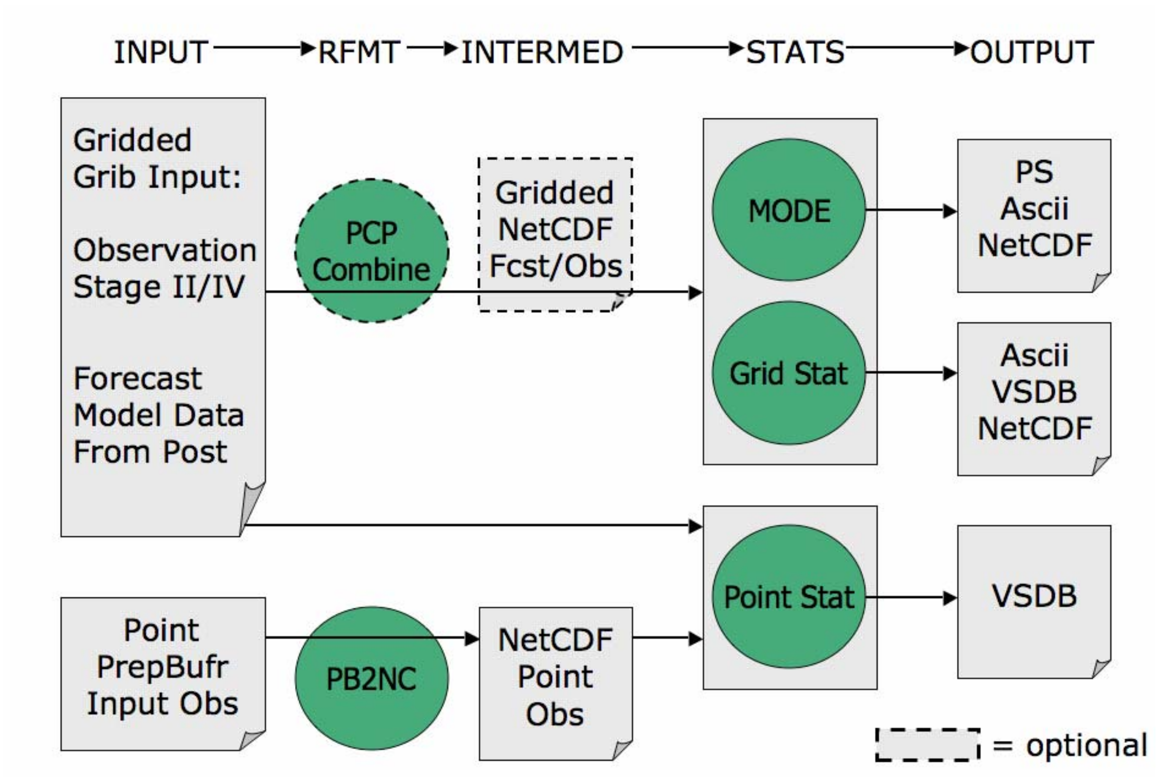


***Figure 1-1**. Basic representation of current MET structure and modules. Green areas represent software and modules included in MET.*

The PB2NC and Pcp-Combine tools are part of the reformatting stage of MET. The PB2NC tool is used to create NetCDF files from input prepbufr files. The NetCDF files are then used in the statistical analysis step. Unlike PB2NC, the Pcp-Combine step is optional and serves only 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 three main statistical analysis components of the current version of MET are: Point-Stat, Grid-Stat, and MODE. The Point-Stat tool is used for grid-to-point verification, or verification of a gridded NWP 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 for some statistics. Confidence intervals take into account the uncertainty associated with verification statistics due to sampling variability and limitations in sample size. They are a valuable tool for obtaining 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 for some verification statistics.

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 field. 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 impossible to obtain using traditional verification measures. For example, the MODE tool can provide information about location, size, and intensity errors.

Results from the statistical analysis stage are output in ASCII format, with the exception of MODE, which also provides some output in PostScript format. The VSDB (Verification System DataBase) is a specialized ASCII-formatted file type that can be easily read and used by graphics and analysis software.

## 1.5 Future development plans

MET is an evolving verification software package. New capabilities are planned in controlled, successive version releases. Since this is a beta release, bug fixes and user-identified problems are expected and will be addressed as they are found. However, plans are in place to incorporate many new capabilities and options in future releases of MET. Some of the planned additions are listed below.

***Additional statistical capabilities***

- Additional spatial forecast verification methods
- Statistics for multi-category contingency tables
- Support for probability forecast and ensemble forecast verification

***Support for other input formats***

- Observations in ASCII and other formats
- Support for forecasts in GRIB2

***Additional analysis capabilities and plotting routines***

- Incorporation of the AFWA GO index
- Aggregation of statistics over time
- Confidence intervals based on resampling procedures
- Confidence intervals based on aggregated statistics
- Additional plotting routines written in IDL, R, and/or NCL that may include, but not be limited to:
    - Boxplots
    - Discrimination plots
    - Reliability diagrams
    - Scatter/density plots
    - Color-fill/contour maps of statistics
    - Height series
    - Histograms

In addition to the items listed above, a long-term goal for MET development is to incorporate MET capabilities in a larger system structure that would include a database for model and observation storage and input/output, as well as a user interface.

## 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 (http://www.dtcenter.org/met/users/).

In this beta release, we welcome all comments and suggestions for improvements to MET, and especially information regarding errors. In addition, comments on this document would be greatly appreciated. While we cannot promise to incorporate all suggested changes (due to funding limitations and other priorities), we will certainly take all suggestions into consideration.

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. To investigate contributing code to the MET, please contact met_help@ucar.edu.

# Chapter 2 – Software Installation/Getting Started

## 2.1 Introduction

The MET package requires three external libraries to be available on the user's computer prior to installation. The initial release of MET has been developed and tested on a single architecture using two sets of compilers. However, support for additional platforms and compilers will be added in future releases.

Three external libraries are required for compiling/building MET:

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 http://www.nco.ncep.noaa.gov/sib/decoders/BUFRLIB. BUFRLIB requires C and Fortran-77 compilers which should be from the same family of compilers used when building MET.

2. Unidata's **NetCDF** libraries are used by several tools within MET for writing output NetCDF files. NetCDF is distributed and supported by Unidata and is 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.

3. The **GNU Scientific Library (GSL) Developer's Version** 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. When installing GSL, be sure to install the Developer's Version because MET requires access to the source headers and library archive file at build time.

Three additional utilities are strongly recommended for use with MET:

1. The **WRF Post-Processor** is recommended for use in post-processing the raw model output prior to verifying the model forecasts with MET. The WRF Post-Processor is freely available for download from the downloads section of the WRF-NMM user's website at http://www.dtcenter.org/wrf-nmm/users. MET requires input data in GRIB1 format on a standard, de-staggered grid and on pressure or regular levels in the vertical. The WRF Post-Processor outputs model data in this format. However, the WRF Post-Processor is not strictly required as long as the user can produce GRIB input data 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 use in re-gridding model and observation datasets in GRIB format to a common verification grid. Prior to running MET, the

model ouput and verifying gridded observations must be placed on the same grid. The copygb utility is distributed as part of the WRF Post-Processor and is available from other sources as well. However, the copygb utility is not strictly required as long as users can ensure that their model and gridded observation datasets reside on a common grid.

3. NCEP's **cwordsh** utility is used to perform Fortran blocking on PrepBufr files containing point observations prior to processing them through MET. The cwordsh utility is distributed and supported by NCEP and is freely available for download from NCEP's website at http://www.nco.ncep.noaa.gov/sib/decoders/BUFRLIB/toc/cwordsh.

## 2.2 Supported architectures

The MET package was developed and tested using the first configuration listed in Table 2-1 for GNU compilers. The MET package has also been built and tested using the Portland Group compilers. Other machines will 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.

***Table2- 1.*** *Hardware and compiler configurations tested for the MET package.*

| **Vendor** | **Hardware** | **OS** | **Compiler** |
|---|---|---|---|
| DELL | XEON | Debian Linux | GNU gcc/f77 |
| DELL | XEON | Debian Linux | PGI pgCC/pgf77 |

The MET package runs on a single processor and there are currently no plans to run it across multiple processors in the future. Therefore, none of the utilities necessary for running WRF on multiple processors are necessary for running MET.

## 2.3 Programming languages

The MET package 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 verification tool 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. Changes to existing tools may be made in their native language of C/C++ or as a Fortran subroutine to be called from the main program. Entirely new tools may be developed by contributors and users using a variety of programming/scripting languages (i.e., C/C++, Fortran, IDL, R, NCL etc.).

## 2.4 Required compilers and scripting languages

The MET package was developed and tested using the GNU `gcc/f77` compilers. The MET package has also been built and tested using the Portland Group `pgCC/pgf77` compilers. As additional compilers are successfully tested, they will be added to the list of supported platforms/compilers.

The Unix `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 include command line utilities as well.

In order to control the desired flow through MET, users are strongly encouraged to run the tools via a script (see Chapter 7 for some examples). 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 to download

As described in the introduction, three 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 gcc/f77 compilers, users are strongly encouraged to use the `-DUNDERSCORE` and `-fno-second-underscore` options. Also, MET expects the BUFRLIB archive file to be named “`libbufr.a`”. Therefore, compiling the BUFRLIB using the GNU `gcc/f77` compilers consists of the following 3 steps:

   - **`gcc -c -DUNDERSCORE -fno-second-underscore *.c`**
   - **`f77 -c -DUNDERSCORE -fno-second-underscore *.f`**
   - **`ar crv libbufr.a *.o`**

   Alternatively, compiling the BUFRLIB using the PGI `pgcc/pgf77` (C and Fortran-77) compilers consists of the following 3 steps:

   - **`pgcc -c -DUNDERSCORE -Mnosecond_underscore *.c`**

- **`pgf77 -c -DUNDERSCORE -Mnosecond_underscore *.f`**
- **`ar crv libbufr.a *.o`**

NOTE: Errors may be encountered when compiling `ufbpos.f`. Since the routines within this file are not called by MET, the errors may be ignored. This problem has been brought to the attention of the BUFRLIB providers.

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.

3. The **GNU Scientific Library (GSL) Developer's Version** is used by MET when computing confidence intervals. When installing GSL, be sure to install the Developer's Version. MET requires access to the source headers and library archive file at build time.

## 2.6 Optional utilities

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

1. The **WRF Post-Processor** is recommended for use in post-processing the raw model output prior to verifying the data with MET. The WRF Post-Processor may be used on output from both the ARW and NMM cores. Please refer to online documentation for instructions on how to install and use the WRF Post-Processor.

2. The **`copygb`** utility is recommended for use in re-gridding model and observation datasets in GRIB format to a common verification grid. The copygb utility is distributed as part of the WRF Post-Processor and is available from other sources as well. Please refer to online documentation for instructions on how to install and use the copygb utility.

3. NCEP's **`cwordsh`** utility is used to perform Fortran blocking on PrepBufr files containing point observations prior to processing them through MET. Once the `cwordsh` tarball has been downloaded and unpacked, refer to the `README_cwordsh` file for instructions on compiling and using this utility. Fortran blocking must be performed on the PrepBufr files prior to reading them with the MET's **pb2nc** utility.

## 2.7 Building the MET package

Building the MET package consists of three main steps: (1) installing the required libraries, (2) configuring the top-level Makefile, and (3) executing the build.

1. **Install the required libraries**
   Please refer to previous sections for instructions on how to obtain and install the required libraries.

2. **Configure the top-level Makefile**
   - Once you have downloaded the MET tarball, unzip and unpack its contents.
   - Edit the top-level **`Makefile`** (or **`Makefile_pgi`**, if using the PGI compilers) as follows:
     - ✧ Set **`CPP_COMPILER`** to the full path for your C/C++ compiler.
     - ✧ Set **`F77_COMPILER`** to the full path for your Fortran-77 compiler.
     - ✧ Set **`MET_BASE`** to the full path for the top level of your MET installation directory.
     - ✧ **`NETCDF_BASE`** may be set to the top level of your NetCDF library for use in defining the next two variables.
     - ✧ Set **`NETCDF_INCS`** to "`-I`" followed by the full path to the directory in your NetCDF library containing the C/C++ header files (ending in .h or .hh).
     - ✧ Set **`NETCDF_LIBS`** to "`-L`" the full path for the directory in your NetCDF library containing the C/C++ library archive files (`libnetcdf.a` and `libnetcdf_c++.a`).
     - ✧ **`BUFR_BASE`** may be set to the top level of your BUFRLIB library for use in defining the next two variables.
     - ✧ Set **`BUFR_INCS`** to "`-I`" followed by the full path to the directory in your BUFRLIB library containing header files (ending in .h or .hh).
     - ✧ Set **`BUFR_LIBS`** to "`-L`" followed by the full path for the directory in your BUFRLIB library containing the library archive file called `libbufr.a`.
     - ✧ If GSL is installed in a non-standard location on your machine, set **`GSL_INCS`** to "`-I`" followed by the full path to the directory in your GSL library containing header files, and set **`GSL_LIBS`** to "`-L`" followed by the full path for the directory in your GSL library containing the library archive file called `libgsl.a`.

3. **Execute the build**
   - Execute the following `make` command to build the MET package, if using GNU compilers:
     - ✧ **`make >& make.log&`**
   - If using the PGI compilers, execute the following "make" command:
     - ✧ **`make –f Makefile_pgi >& make.log&`**
   - Execute the following "tail" command to monitor the progress of the make:
     - ✧ **`tail -f make.log`**
   - When the make has completed, use CNTL-F to end the tail command.

- Examine the contents of the **`make.log`** file.
  - ✧ Look for the following message which likely indicates that the build was successful:

```
*** Finished Making the Model Evaluation Tools Project ***
```

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

## 2.8 MET directory structure

The top-level MET directory structure is illustrated by the following outline:

```
MET/
  •  Makefile
  •  Makefile_pgi
  •  README
  •  bin/
       o  grid_stat          mode           pb2nc
       o  pcp_combine        point_stat
  •  data/
       o  colortables/       map/           ps/
       o  poly/              config/        sample_fcst/
       o  sample_obs/
  •  doc/
       o  Documentation
  •  lib/
       o  Several Internal Libraries
  •  out/
       o  Sample Output
  •  scripts/
       o  Sample Scripts
       o  config/
  •  src/
       o  pcp_combine/       grid_stat/     mode/
       o  pb2nc/             point_stat/
```

The top-level MET directory consists of a README file, two Makefiles, and several subdirectories. The top-level Makefiles control how the entire toolkit is built by calling sub-makes for each of the internal libraries and applications. These top-level Makefiles were modified in Section 2.7.

When MET has been successfully built, the `bin/` directory contains executables for each module of MET (**`grid_stat`**, **`mode`**, **`pb2nc`**, **`pcp_combine`**, and **`point_stat`**)

The **data/** directory contains several configuration and static data files used by MET. The colortables/, map/, and ps/ subdirectories contain data used in creating PostScript plots for the MODE tool. The poly/ directory 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 sample configuration files for each MET tool that accepts one. Users may copy these configuration files to another location and modify them for their own use. 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 **lib/** directory contains the source code for several internal libraries used by MET tools.

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 five tools in MET.

The **scripts/** directory contains test scripts to be run after MET has been successfully built. , as well as a directory of sample configuration files located in scripts/config. The output from the test scripts in this directory, scripts/, 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.

## 2.9 Sample test cases

Once the MET package has been built successfully, the user is encouraged to run the sample test scripts provided. Change directories into the **scripts/** directory. The scripts directory contains one test Bourne shell script for each of the five tools in MET. However, the **test_all.sh** script will run the other five scripts in the proper order. Execute the following commands:

- Run the script.
```
./test_all.sh >& test_all.log&
```

- Monitor the progress of the script:
```
tail -f test_all.log
```

- When the test script has completed, use CNTL-F to end the tail command.

  NOTE: Most of the steps in the test scripts will only take a few seconds each to run. However, the PB2NC utility step will likely take a few minutes.

- Examine the contents of the **`test_all.log`** file:
    - ✧ Look for the following message which indicates that the test script completed:

```
*** Finished Testing the Model Evaluation Tools Project ***
```

    - ✧ If any warning or errors occur, please refer to Appendix A on troubleshooting for common problems.

- The output from this test script is written to the top-level `out/` directory, organized by the name of the MET tool run.

# Chapter 3 – MET Data I/O and Re-Formatting

Both input and output file formats are considered 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 Fig. 1-1. Output data formats and the software modules used to reformat the data are considered in later sections.

## 3.1 Input data formats

The MET package can handle input data in two formats: GRIB version 1 (i.e., the same as the output format produced by the WRF Post-Processor) and PrepBufr for point observation files which are consistent with the existing NCEP verification tools. Note that MET does not require the WRF 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. Some two-dimensional fields, such as precipitation, can also be accepted in some modules such as the Grid-Stat and MODE tools.

When comparing two gridded fields with the Grid-Stat or MODE tools, the input model and observation datasets must have already been placed on the same grid. The `copygb` utility is recommended for use in re-gridding GRIB files. 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 observation types in a single file in a standard format. However, some users may desire 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. Users may convert observations that are not available in the PrepBufr files into this NetCDF format for use by the Point-Stat verification tool. In future releases, consideration will be given to adding greater flexibility for the point observation input formats, such as allowing for formatted ASCII files.

## 3.2 Intermediate data formats

MET uses NetCDF as an intermediate file format. Both the Pcp-Combine and PB2NC tools write intermediate files in NetCDF format.

The Pcp-Combine tool is used to sum accumulated precipitation from several GRIB files into a single NetCDF file containing the desired accumulation period. The user may choose to: (1) sum the model accumulations to match the observation accumulation period, (2) sum the observation accumulations to match the model accumulation period, or (3) sum both the model and observation accumulations to some new period desired for verification. In performing this summation, the user may not specify an accumulation interval smaller than the accumulation period in the GRIB files. However, if the input model and observation GRIB files already contain accumulated precipitation with the same desired accumulation period, `pcp_combine` need not be run. Each time the Pcp-Combine tool is called, a NetCDF file is written containing the requested accumulation period.

The PB2NC tool is used to reformat the input PrepBufr files containing point observations. `pb2nc` stratifies the observations as requested in a configuration file and writes out the remaining observations in a NetCDF format. The NetCDF output of the PB2NC tool is used as input to the verification step performed in the Point-Stat tool.

## 3.3 Output data formats

The MET package currently produces output in four basic file formats: VSDB files, ASCII files, gridded NetCDF files, and PostScript plots.

The VSDB (Verification Statistics Database) format was chosen as an output file type to be consistent with the existing verification tools used by NCEP. NCEP uses VSDB files to populate a database containing verification statistics for the models they run. MET produces VSDB output for both the Grid-Stat and Point-Stat tools. VSDB is simply a specialized ASCII format containing one record on each line. However, a single VSDB file may contain multiple line types. The first group of columns before the equal sign delimiter remains the same for each line type. However, the meaning for the columns after the equal sign delimiter changes for each line type. Therefore, VSDB files can be difficult for a human to read as the meaning for many columns of data changes from line to line.

For this reason, ASCII output is also available as an alternative for the Grid-Stat and Point-Stat tools. The ASCII files contain exactly the same output as the VSDB files but each VSDB 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 also creates two ASCII output files, although they are not related to a VSDB file. The MODE tool 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.

Both the Grid-Stat and the MODE tool generate gridded NetCDF output. The MODE tool creates a NetCDF file containing four gridded fields for the indices of the forecast and observation simple and composite object fields. The Grid-Stat tools creates a NetCDF file containing the forecast minus observation difference fields for each verification region and variable type/level requested in the configuration file. In both cases, the generation of these files is controlled by configuration files. As discussed in the previous section, both the Pcp-Combine and PB2NC tools create intermediate NetCDF files as well.

The MODE tool produces 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. It contains four 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 observation fields. The generation of this PostScript output can be disabled using a command line option.

## 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 that has been Fortran-blocked, and one configuration file.
   - **Output**: One NetCDF file containing the observations that have been retained.

2. **Point-Stat Tool**
   - **Input**: One model file in GRIB format, one point observation file in NetCDF format (as the output of the PB2NC tool), and one configuration file.
   - **Output**: One VSDB file containing all of the requested line types, and several ASCII files for each line type requested.

3. **Pcp-Combine Tool**
   - **Input**: Several gridded model or observation files in GRIB format containing accumulated precipitation to be summed to create a new accumulation interval.
   - **Output**: One NetCDF file containing the summed accumulation interval.

4. **Grid-Stat Tool**
    - **Input**: One model file and one observation file either in GRIB format or in NetCDF format (as the output of the Pcp-Combine tool), and one configuration file.
    - **Output**: One VSDB file containing all of the requested line types, several ASCII files for each line type requested, and one NetCDF file containing the difference field for each verification region and variable type/level being verified.

5. **MODE Tool**
    - **Input**: One model file and one observation file either in GRIB format or in NetCDF format (as the output of the Pcp-Combine tool), 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 indices for the gridded simple and composite object fields, and one PostScript plot containing a summary of the features-based verification performed.

## 3.5 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 the Point-Stat tool. The PB2NC tool must be run on the input PrepBufr point observation file prior to performing verification using the Point-Stat tool. In addition, prior to running the PB2NC tool, the input PrepBufr file will likely need to be Fortran-blocked using the `cwordsh` utility. Fortran-blocking has already been performed on the test data distributed with MET, so no additional blocking should be necessary for these data. However, when running the Point-Stat tool with other datasets, Fortran-blocking will likely be necessary.

It is the user's responsibility to download and install the `cwordsh` utility, as described in Chapter 2. The usage statement for the `cwordsh` utility is listed below:

```
cwordsh   action   inputfile      outputfile
```

The **`action`** argument can be set to **`block`** or **`unblk`** to indicate whether blocking or unblocking is to be performed. **`inputfile`** is the name of the input PrepBufr file, and **`outputfile`** is the file name to be created.

An example of an application of the `cwordsh` utility to perform Fortran-blocking is shown below:

```
cwordsh   block    sample_pb      sample_pb.blk
```

In this example, Fortran-blocking will be performed on the input **`sample_pb`** PrepBufr file and the output will be written to **`sample_pb.blk`**.

### *3.5.1* `pb2nc` *usage*

Once the input file has been Fortran-blocked, the PrepBufr file is ready to be processed by the PB2NC tool. The usage statement for the PB2NC tool is shown below:

```
Usage: pb2nc        prepbufr_file       netcdf_file
                    [ -config config_file ]
                    [ -v level ]
                    [ -nmsg num_messages ]
                    [ -dump dump_dir ]
```

`pb2nc` has two required arguments and can take up to four optional ones.

***Required arguments for*** **`pb2nc`**

1. The **prepbufr_file** argument indicates the name of the Fortran-blocked PrepBufr file to be processed.

2. The **netcdf_file** argument indicates the name given to the output NetCDF file.

***Optional arguments for*** **`pb2nc`**

1. The **`-config`** **config_file** option indicates the name of the configuration file to be used instead of the default configuration. The contents of the configuration file are discussed below.

2. The **`-v`** **level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 1. 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.

3. 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.

4. The **`-dump`** **dump_dir** option may be used to dump the entire contents of the PrepBufr file to several ASCII files written to the directory specified by "**dump_dir**". 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.

An example of the `pb2nc` calling sequence is shown below:

```
pb2nc          sample_pb.blk  sample_pb.nc
               -config PB2NCConfig
```

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

### *3.5.2* `pb2nc` *configuration file*

The default configuration file for the PB2NC tool named **`PB2NCConfig_default`** can be found in the `data/config` directory in the MET distribution. The version used for the example run in Chapter 2 is also available in `scripts/config`. It is recommended that users make a copy of these files prior to modifying their contents, and use the `-config` option when specifying configuration files other than the default. Each configuration file contains many comments describing its contents.

The contents of the default `pb2nc` configuration file found in `data/config` are described in the subsections below.

---

```
message_type = "";
```

Each PrepBufr message is tagged with one of eighteen message types as listed in the configuration file. The user may specify a space separated list of message types to be retained. Providing an empty list indicates that all message types should be retained.

---

```
station_id = "";
```

Each PrepBufr message has a station identification string associated with it. The user may specify a space separated list of station IDs to be retained. Providing an empty list indicates that messages from all station IDs will be retained.

---

```
beg_ds = -5400;
end_ds =  5400;
```

The PrepBufr file has an observation time associated with it. Each PrepBufr message within the file has a time offset defined relative to that file observation time. The **`beg_ds`** and **`end_ds`** variables define a time window around the file's observation time for PrepBufr messages that should be retained. The time offset is given in seconds. By default, the time window is +/- 1.5 hours (= 5400 seconds) around the file observation time.

---

```
mask_grid = "";
mask_poly = "";
```

The **`mask_grid`** and **`mask_poly`** variables are used to define a spatial masking region for retaining observations. **`mask_grid`** may be set to one of the pre-defined NCEP grids which are specified as **GNNN** where **NNN** is the three digit designation for the grid. **`mask_poly`** may be set to a pre-defined or a user-created file consisting of a name for the polygon followed by a series of lat/lon points used to define a masking region. If a masking region is specified, only observations falling inside the region will be retained. Refer to Appendix B for a list of the grids available for `mask_grid` and pre-defined polylines for `mask_poly`.

---

```
beg_elev = -1000;
end_elev = 100000;
```

The **`beg_elev`** and **`end_elev`** variables are used to stratify the elevation of the observations to be retained. The units are in meters and by default the range 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 space 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.

---

```
beg_level = 1;
end_level = 255;
```

The **`beg_level`** and **`end_level`** variables are used to stratify the model level of observations to be retained. The default range is 1 to 255, which is the current maximum possible level.

---

```
obs_grib_code = "PRES SPFH TMP HGT UGRD VGRD";
```

Each PrepBufr message will likely contain multiple observation variables. The **`obs_grib_code`** variable is used to specify which observation variables are to be retained. The GRIB code itself or the corresponding abbreviation may be used to specify which observation variables are to be retained. By default, pressure, specific humidity, temperature, height, and the u and v components of the wind are retained.

In the initial release of MET, no new observation variables are derived from these basic ones. However, in future releases, the derivation of new observation variables will be supported.

---

**`quality_mark_threshold = 2;`**

Each observation has a quality mark value associated with it. The **`quality_mark_threshold`** is used to stratify out which quality marks will be retained. By default, only observation with quality marks less than or equal to 2 will be retained.

---

**`multiple_quality_marks_flag = 0;`**

The PrepBufr message may contain duplicate observations with different quality mark values. The **`multiple_quality_marks_flag`** indicates whether multiple versions of observations with different quality marks should be retained. By default, the flag value of 0 indicates that only the highest quality version of the observation should be retained.

---

**`level_category = "";`**

Lastly, the **`level_category`** variable is used to specify a space-separated list of data level categories to retain. An empty string indicates that all level categories should be retained. See the configuration file for more details.

---

### *3.5.3 PB2NC output*

Each NetCDF file generated by the PB2NC tool contains the dimensions and variables shown in the following tables.

| **`pb2nc`** ***NetCDF DIMENSIONS*** | |
|---|---|
| **NetCDF Dimension** | **Description** |
| `mxstr` | Maximum string length (20) |
| `hdr_arr_len` | Number of entries in each PrepBufr message header array (7) |
| `obs_arr_len` | Number of entries in each PrepBufr observation array (11) |
| `nobs` | Number of PrepBufr observations in the file (UNLIMITED) |
| `nmsg` | Number of PrepBufr messages in the file (variable) |

| `pb2nc` ***NetCDF VARIABLES*** | | |
|---|---|---|
| **NetCDF Variable** | **Dimension** | **Description** |
| `obs_arr` | nobs, obs_arr_len | Array of floats containing values for each observation including:<br>• Reference to the entry in the hdr_arr with which this observation is associated<br>• Vertical level<br>• Pressure level in hPa or mb<br>• GRIB code corresponding to this observation type<br>• Observation value<br>• Quality mark<br>• Program code<br>• Reason code<br>• Forecast value<br>• Analyzed value<br>• Data level category |
| `hdr_typ` | nmsg, mxstr | Text string containing the message type for each PrepBufr message. |
| `hdr_sid` | nmsg, mxstr | Text string containing the station id for each PrepBufr message. |
| `hdr_vld` | nmsg, mxstr | Text string containing the observation valid time for each PrepBufr message in YYYY-MM-DD:HH:MM:SS in UTC format. |
| `hdr_arr` | nmsg, hdr_arr_len | Array of floats containing values for each PrepBufr message including:<br>• Longitude in degrees east<br>• Latitude in degrees north<br>• Time offset in fractional hours from file valid time<br>• Elevation in meters<br>• PrepBufr report type<br>• Input report type<br>• Instrument type |

## 3.6 Pcp-Combine tool

This section contains a description of running the Pcp-Combine tool. The Pcp-Combine tool is used (if needed) to sum accumulated precipitation from several GRIB files into a single NetCDF file containing the desired accumulation period, for input to the MODE and Grid-Stat tools. Currently, the Pcp-Combine tool will sum only accumulated precipitation, GRIB code 61. The GRIB files being combined must have already been placed on the grid on which the user would like to verify. The `copygb` utility is recommended for use in re-gridding GRIB files. In addition, the Pcp-Combine tool will only sum model files with the same initialization time unless directed to ignore the initialization time.

### *3.6.1* `pcp_combine` *usage*

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

```
Usage: pcp_combine  pcp_init_time  pcp_accum_period
                    valid_time     valid_accum_period
                    out_file
                    [ -pcpdir precip_dir ]
                    [ -pcprx precip_reg_exp ]
                    [ -v level ]
```

`pcp_combine` has five required arguments and accepts up to three optional ones.

#### *Required arguments for* `pcp_combine`

1. The **pcp_init_time** argument, provided in YYYY-MM-DD_HH:MM:SS 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 (0000-00-00_00:00:00) indicates that all files, regardless of initialization time should be processed.

2. The **pcp_accum_period** argument, provided in HH format, indicates the accumulation interval of the model or observation GRIB files that are being 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 YYYY-MM-DD_HH:MM:SS format, indicates the desired valid time to which the accumulated precipitation is to be summed.

4. The **valid_accum_period** argument, in HH format, indicates the desired total accumulation period to be summed.

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

#### *Optional arguments for* `pcp_combine`

1. The **`-pcpdir`** **precip_dir** option indicates the directory in which the input GRIB files reside. The contents of “precip_dir” will override the default setting.

2. The **`-pcprx`** **precip_reg_exp** option indicates the regular expression to be used in matching files in the precipitation directory specified. The contents of “**precip_reg_exp**” will override the default setting which matches all file names.

If the precipitation directory contains 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.

3. The **`-v` level** option indicates the desired level of verbosity. The contents of “level” will override the default setting of 1. 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.

An example of the `pcp_combine` calling sequence is presented below:

<u>Example 1:</u>

```
pcp_combine    2005-08-07:00_00:00 3
               2005-08-08_00:00:00 24
               sample_fcst.nc
               -pcpdir ../data/sample_fcst/2005080700
```

In Example 1, the Pcp-Combine tool is instructed to combine 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 GRIB 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:

<u>Example 2:</u>

```
pcp_combine    0000-00-00:00_00:00 1
               2005-08-08_00:00:00 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 “**pcp_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 “**pcp_accum_period**” has been changed from 3 to 1 to indicate that the input GRIB observation files contain 1-hourly accumulations of precipitation. Lastly, **`-pcpdir`** provides a different directory to be searched for the input GRIB 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.

### 3.6.2 `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 Grid-Stat and MODE tools in subsequent steps. One may use commonly available NetCDF utilities such as `ncdump` or `ncview` to view the contents of the output file.

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

| `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) |

| `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 |
| `precip` | `lat, lon` | Amount of accumulated precipitation for each point in the grid |

# Chapter 4 – The Point-Stat Tool

## 4.1 Introduction

The Point-Stat tool provides verification statistics for forecasts at points (as opposed to the other alternative, across grids). The Point-Stat tool matches gridded forecasts to point observation locations, using several different interpolation approaches. The tool then computes continuous as well as categorical verification statistics. The categorical 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 some of the verification statistics

Scientific and statistical aspects of the Point-Stat tool are considered in the following section. Practical aspects of using the Point-Stat tool are considered in Section 4.3.

## 4.2 Scientific and statistical aspects

The statistical methods and measures computed by the Point-Stat tool are described in this section. In addition, Section 4.2.1 considers the various interpolation options that are available for matching the forecast gridpoint values to the observation points. The statistical measures computed by the Point-Stat tool are described in Section 4.2.2, and Section 4.2.3 describes methods for computing confidence intervals that are applied to some of the measures computed by the Point-Stat tool.

### *4.2.1 Interpolation methods*

This section provides information about the various interpolation methods available in MET, and applied in the Point-Stat tool to match gridded model output to point observations. In these descriptions, we'll use P to denote the point where the interpolated value is being matched. An interpolation width W also needs to be specified. For example, a width of 2 means a 2 x 2 square enclosing P. Any vertical interpolation needed is done in natural log of pressure coordinates. This section describes the different options for interpolation in the horizontal.

The methods are briefly described in the following subsections.

**Nearest Neighbor**

No interpolation is performed. The data value at P is simply the value at the nearest grid point. Here, "nearest" means nearest in grid coordinates. This method is used by default when the interpolation width W is set to 1.

**Minimum value**

The data value at P is the minimum of the data values in the W x W square.

**Maximum value**

The data value at P is the maximum of the data values in the W x W square.

**Distance-weighted mean**

The data value at P is a weighted sum of the values in the W x W square, the weight at each such point being the reciprocal of the square of the distance (in grid coordinates) from P. The weighted sum of data values is then normalized by dividing by the sum of the weights. As a special case, bilinear interpolation (see below) is used if W=2.

**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 does not matter.

**Median**

The value at P is the median of the data values in the W x W square.

**Bilinear interpolation**

The bilinear interpolation method has two steps, each of which consists of one or more simple linear interpolations.

First, a linear interpolation is performed on the upper and lower left corners of the 2 x 2 square to get an interpolated value on the left side of the square. Similarly, a value on the right side of the square is calculated by a linear interpolation of the upper right and lower right data values. Finally, a value at P is interpolated from the right and left-hand interpolated values.

Although we have described this process as vertical interpolations followed by horizontal interpolations, it turns out the order doesn't matter. The same result would be obtained if horizontal interpolations were performed first, followed by a vertical interpolation.

### *4.2.2 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 measures are briefly described here; more information on them can be found in Wilks (2006) 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 the partial sums and other scores that are produced by the NCEP verification system. These statistics are also described in this section.

*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. Currently, Point-Stat computes statistics for variables in two categories. In future versions, MET will include the capability to compute measures for multi-category forecasts. The categories for dichotomous (i.e., 2-category) variables can be intrinsic (e.g., rain/no-rain) or they may be formed by applying a threshold to a continuous variable (e.g., temperature < 32°F).

The verification statistics for dichotomous variables are formulated using a contingency table such as the one shown in Table 4-1. 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 4-1 are counts of the number of occurrences of the four possible combinations of forecasts and observations.

***Table 4-1**: 2x2 contingency table in terms of counts. The $n_{ij}$ 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")** | $n_{11}$ | $n_{10}$ | $n_{1.} = n_{11} + n_{10}$ |
| **$f$ = 0 (e.g., "No")** | $n_{01}$ | $n_{00}$ | $n_{.0} = n_{01} + n_{00}$ |
| **Total** | $n_{.1} = n_{11} + n_{01}$ | $n_{.0} = n_{10} + n_{00}$ | $T = n_{11} + n_{10} + n_{01} + n_{00}$ |

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, $p_{11}$, $p_{10}$, $p_{01}$, and $p_{00}$ can be computed. Note that $p_{11} + p_{10} + p_{01} + p_{00} = 1$. 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 4-1 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 tool are described in the following subsections.

---

**Base rate**

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., $o = 1$). The base rate is equivalent to the "O" value produced by the NCEP Verification System.

---

**Mean forecast**

The mean forecast value is defined as $\bar{f} = \frac{n_{11} + n_{10}}{T} = \frac{n_{1.}}{T}$. This statistic is comparable to the base rate and is the relative frequency of occurrence of a forecast of the event (i.e., $f = 1$). The mean forecast is equivalent to the "F" value computed by the NCEP Verification System.

---

**Accuracy**

Accuracy for a 2x2 contingency table is defined as $\frac{n_{11} + n_{00}}{T}$. 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 $n_{00}$.

---

**Frequency Bias**

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 $\text{Bias} = \frac{n_{11} + n_{10}}{n_{11} + n_{01}} = \frac{n_{1.}}{n_{.1}}$. 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)

POD is defined as $\mathrm{POD} = \frac{n_{11}}{n_{11} + n_{01}} = \frac{n_{11}}{n_{.1}}$. 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)

POFD is defined as $\mathrm{POFD} = \frac{n_{10}}{n_{10} + n_{00}} = \frac{n_{10}}{n_{.0}}$. It is the proportion of non-events that were forecast to be events. POFD is also often called the **False Alarm Rate**[1]. POFD ranges from 0 to 1; a perfect forecast would have POFD = 0.

## Probability of Detection of the non-event (PODn)

PODn is defined as $\mathrm{PODn} = \frac{n_{00}}{n_{10} + n_{00}} = \frac{n_{00}}{n_{.0}}$. 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)

FAR is defined as $\mathrm{FAR} = \frac{n_{10}}{n_{11} + n_{10}} = \frac{n_{10}}{n_{1.}}$. It is the proportion of forecasts of the event occurring for which that the event did not occur. FAR ranges from 0 to 1; a perfect forecast would have FAR = 0.

## Critical Success Index (CSI)

CSI is defined as $\mathrm{CSI} = \frac{n_{11}}{n_{11} + n_{10} + n_{01}}$. 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., $n_{00}$). 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.

[1] Note that the false alarm **rate** is not the same as the false alarm **ratio**, also described here.

**Gilbert Skill Score (GSS)**

GSS is based on the CSI, corrected for the number of hits that would be expected by chance. In particular, $\mathrm{GSS} = \dfrac{n_{11} - C_1}{n_{11} + n_{10} + n_{01} - C_1}$, where $C_1 = \dfrac{(n_{11} + n_{10})(n_{11} + n_{01})}{T} = \dfrac{n_{1.} n_{.1}}{T}$. 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)**

H-K is defined as $\text{H-K} = \dfrac{n_{11} n_{00} - n_{10} n_{01}}{(n_{11} + n_{01})(n_{10} + n_{00})}$. More simply, $\text{H-K} = \mathrm{POD} - \mathrm{POFD}$. 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)**

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, $\mathrm{HSS} = \dfrac{n_{11} + n_{00} - C_2}{T - C_2}$, where $C_2 = \dfrac{(n_{11} + n_{10})(n_{11} + n_{01}) + (n_{01} + n_{00})(n_{10} + n_{00})}{T}$. HSS can range from minus infinity to 1. A perfect forecast would have HSS = 1.

**Odds Ratio (OR)**

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 $\mathrm{OR} = \dfrac{n_{11} \times n_{00}}{n_{01} \times n_{10}} = \dfrac{\left(\dfrac{\mathrm{POD}}{1 - \mathrm{POD}}\right)}{\left(\dfrac{\mathrm{POFD}}{1 - \mathrm{POFD}}\right)}$.

OR can range from 0 to infinity. 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).

*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**
**Mean observation**

The sample mean forecast and the sample mean observation are defined as

$$\overline{f} = \frac{1}{n}\sum_{i=1}^{n} f_i \text{ and } \overline{o} = \frac{1}{n}\sum_{i=1}^{n} o_i \text{, respectively.}$$

---

**Forecast standard deviation**
**Observation standard deviation**

The sample variance of the forecasts and observations are defined as

$$s_f^2 = \frac{1}{n-1}\sum_{i=1}^{T}(f_i - \overline{f})^2 \text{ and } s_o^2 = \frac{1}{n-1}\sum_{i=1}^{n}(o_i - \overline{o})^2 .$$

The forecast and observed standard deviations are simply defined as

$$s_f = \sqrt{s_f^2} \text{ and } s_o = \sqrt{s_o^2} .$$

---

**Pearson Correlation Coefficient**

The Pearson correlation coefficient, $\boldsymbol{r}$, measures the strength of linear association between the forecasts and observations. The Pearson correlation coefficient is defined as

$$r = \frac{\sum_{i=1}^{n}(f_i - \overline{f})(o_i - \overline{o})}{\sqrt{\sum_{i=1}^{n}(f_i - \overline{f})^2}\sqrt{\sum_{i=1}^{n}(o_i - \overline{o})^2}}$$

$r$ 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. MET provides confidence intervals for $r$.

---

**Mean Error (ME)**

The Mean Error, **ME**, is a measure of overall bias for continuous variables. It is defined as

$$\mathrm{ME} = \frac{1}{n}\sum_{i=1}^{n}(f_i - o_i) = \overline{f} - \overline{o} \ .$$

A perfect forecast has **ME** = 0.

---

**Mean-squared error (MSE)**
**Root-mean-squared error (RMSE)**
**Standard deviation of the error**

The **MSE** measures the average squared error of the forecasts. Specifically,

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(f_i - o_i)^2 \ .$$

**RMSE** is simply the square root of the **MSE**, $\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$ .

**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

$\mathrm{MSE} = (\overline{f} - \overline{o})^2 + s_f^2 + s_o^2 - 2s_f s_o r_{fo}$ , where $s_f^2 + s_o^2 - 2s_f s_o r_{fo}$ is the estimated variance of the error, $s_{f-o}^2$ . Thus, $\mathrm{MSE} = \mathrm{ME}^2 + s_{f-e}^2$ . Thus, it is important to examine both of the terms of **MSE**, rather than just **MSE**. Moreover, **MSE** can be strongly influenced by **ME**.

The standard deviation of the error, $s_{f-o}$, is simply $s_{f-o} = \sqrt{s_{f-o}^2} = \sqrt{s_f^2 + s_o^2 - 2s_f s_o r_{fo}}$ .

---

## Mean Absolute Error (MAE)

The **Mean Absolute Error (MAE)** is defined as

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left| f_i - o_i \right|.$$

**MAE** is less influenced by large errors and also does not depend on the mean error. A perfect forecast would have **MAE** = 0.

## Percentiles of the errors

**Percentiles of the errors** provide greater 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.

## Scalar L1 and L2 values

These statistics (included in the SL1L2 output file) are simply the 1st and 2nd moments of the forecasts, observations and errors:

$$\text{Mean}(f) = \overline{f} = \frac{1}{n}\sum_{i=1}^{n} f_i \quad \text{Mean}(o) = \frac{1}{n}\sum_{i=1}^{n} o_i \quad \text{Mean}(f*o) = \frac{1}{n}\sum_{i=1}^{n}(f*o)$$

$$\text{Mean}(f^2) = \frac{1}{n}\sum_{i=1}^{n} f^2 \text{ and } \text{Mean}(o^2) = \frac{1}{n}\sum_{i=1}^{n} o^2$$

Some of the other statistics for continuous forecasts can be derived from these moments.

## Scalar anomaly L1L2 values

Computation of these statistics requires a climatological value, $c$. These statistics are the 1st and 2nd moments of the scalar anomalies and are included in the SAL1L2 output file. The moments are defined as:

$$\text{Mean forecast anomaly} = \text{Mean}(f-c) = \frac{1}{n}\sum_{i=1}^{n}(f_i - c) = \overline{f} - c$$

$$\text{Mean observed anomaly} = \text{Mean}(o-c) = \frac{1}{n}\sum_{i=1}^{n}(o_i - c) = \overline{o} - c$$

$$\text{Mean}[(f-c)(o-c)] = \frac{1}{n}\sum_{i=1}^{n}(f_i - c)(o_i - c) \quad \text{Mean}[(f-c)^2] = \frac{1}{n}\sum_{i=1}^{n}(f_i - c)^2$$

$$\text{Mean}[(o-c)^2] = \frac{1}{n}\sum_{i=1}^{n}(o_i - c)^2$$

## Vector L1 and L2 values

These statistics are the moments for wind vector values, where $u$ is the E-W wind component and $v$ is the N-S wind component ($u_f$ is the forecast E-W wind component; $u_o$ is the observed E-W wind component; $v_f$ is the forecast N-S wind component; and $v_o$ is the observed N-S wind component). The following measures are computed:

$$\text{Mean}(u_f) = \frac{1}{n}\sum_{i=1}^{n}u_{fi} = \bar{u}_f \quad \text{Mean}(v_f) = \frac{1}{n}\sum_{i=1}^{n}v_{fi} = \bar{v}_f$$

$$\text{Mean}(u_o) = \frac{1}{n}\sum_{i=1}^{n}u_{oi} = \bar{u}_o \quad \text{Mean}(v_o) = \frac{1}{n}\sum_{i=1}^{n}v_{oi} = \bar{v}_o$$

$$\text{Mean}(u_f u_o + v_f v_o) = \frac{1}{n}\sum_{i=1}^{n}(u_{fi}u_{oi} + v_{fi}v_{oi})$$

$$\text{Mean}(u_f^2 + v_f^2) = \frac{1}{n}\sum_{i=1}^{n}(u_{fi}^2 + v_{fi}^2) \quad \text{Mean}(u_o^2 + v_o^2) = \frac{1}{n}\sum_{i=1}^{n}(u_{oi}^2 + v_{oi}^2)$$

## Vector anomaly L1 and L2 values

These statistics require climatological values for the wind vector components, $u_c$ and $v_c$. The measures are defined below:

$$\text{Mean } u \text{ forecast anomaly} = \text{Mean}(u_f - u_c) = \frac{1}{n}\sum_{i=1}^{n}(u_f - u_c) = \bar{u}_f - u_c$$

$$\text{Mean } v \text{ forecast anomaly} = \text{Mean}(v_f - v_c) = \frac{1}{n}\sum_{i=1}^{n}(v_{fi} - v_c) = \bar{v}_f - v_c$$

$$\text{Mean } u \text{ observation anomaly} = \text{Mean}(u_o - u_c) = \frac{1}{n}\sum_{i=1}^{n}(u_{oi} - u_c) = \bar{u}_o - u_c$$

$$\text{Mean } v \text{ observation anomaly} = \text{Mean}(v_o - v_c) = \frac{1}{n}\sum_{i=1}^{n}(v_{oi} - v_c) = \bar{v}_o - v_c$$

$$\text{Mean}[(u_f - u_c)(u_o - u_c) + (v_f - v_c)(v_o - v_c)] = \frac{1}{n}\sum_{i=1}^{n}[(u_{fi} - u_c)(u_{oi} - u_c) + (v_{fi} - v_c)(v_{oi} - v_c)]$$

$$\text{Mean}[(u_f - u_c)^2 + (v_f - v_c)^2] = \frac{1}{n}\sum_{i=1}^{n}[(u_{fi} - u_c)^2 + (v_{fi} - v_c)^2]$$

$$\text{Mean}[(u_o - u_c)^2 + (v_o - v_c)^2] = \frac{1}{n}\sum_{i=1}^{n}[(u_{oi} - u_c)^2 + (v_{oi} - v_c)^2]$$

### *4.2.3* **100(1 - $\alpha$)%** *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 $\alpha$ are 0.01, 0.05, and 0.10. The Point-Stat tool allows the user to select the specific $\alpha$-value 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.

Future versions of the MET package will include confidence intervals for additional variables, including other skill scores and categorical statistics (e.g., POD, FAR).

For more information on confidence intervals pertaining to verification scores, see Wilks (2006) and Jolliffe and Stephenson (2003).

## 4.3 Practical information

This section contains a description of how to configure and run the Point-Stat tool. 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 GRIB-1 format. The point observations must be in NetCDF format as the output of the `pb2nc` step. 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 4.2.1. 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.

### *4.3.1* `point_stat` *usage*

The usage statement for the Point-Stat tool is shown below:

```
Usage: point_stat   fcst_file obs_file  model
                    [ -config config_file ]
                    [ -climo climo_file ]
                    [ -outdir output_dir ]
                    [ -v level ]
```

`point_stat` has three required arguments and can take up to four optional ones.

#### *Required arguments for* `point_stat`

1. The **fcst_file** argument identifies the GRIB file containing the model data to be verified.

2. The **obs_file** argument indicates the NetCDF file containing the point observations to be used for the verification of the model.

3. The **model** argument is a short text string indicating a name for the model being verified. The “model” text string will be written to the output VSDB and ASCII files. When data from multiple models are combined in a database, the “model” string is used to distinguish the models that were verified.

#### *Optional arguments for* `point_stat`

1. The **`-config`** **config_file** option indicates the name of the configuration file to be used instead of the default configuration. The contents of the configuration file will be discussed shortly.

2. The **`-climo`** **climo_file** identifies the GRIB file containing climatological values on the same grid as the forecast file to be used when computing scalar and vector anomaly measures. If the “climo_file” is not provided, scalar and vector anomaly values will not be computed.

3. The **`-outdir`** **output_dir** indicates the directory where output files should be written.

4. The **`-v`** **level** option indicates the desired level of verbosity. The contents of “level” will override the default setting of 1. 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.

An example of the `point_stat` calling sequence is shown below:

```
point_stat     sample_fcst.grb
               sample_pb.nc
               WRF
               -config 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_pd.nc`** applying the configuration options specified in the PointStatConfig file. The name chosen for the model data is simply `WRF`.

### *4.3.2* `point_stat` *configuration file*

The default configuration file for the Point-Stat tool named `PointStatConfig_default` can be found in the data/config directory in the MET distribution. Another version is located in `scripts/config`. We encourage users to make a copy of these files prior to modifying their contents. The `-config` option must be invoked if using a file other than the default in `scripts/config`. 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.

```
vx_grib_code = "SPFH/P500 TMP/P500 HGT/P500 UGRD/P500 VGRD/P500";
```

The **`vx_grib_code`** variable contains a space-separated list of model variables and corresponding vertical levels to be verified. The GRIB code itself or the corresponding abbreviation may be used to specify which model fields are to be verified. Each GRIB code must be followed by a level indicator in the form "**ANNN**", "**LNNN**", "**PNNN**", or "**PNNN-NNN**" for an accumulation interval, a single vertical level, a single pressure level, or a range of pressure levels. "**NNN**" indicates the accumulation or level value. By default, `point_stat` will verify specific humidity, temperature, height, and the U and V components of the winds, all at 500 mb. All variables are treated as scalar quantities with the exception of the U and V components of the wind. When the U component is followed by the V component, both with the same level indicator, they will be treated as vector quantities. A list of GRIB codes is available at http://www.nco.ncep.noaa.gov/pmb/docs/on388/table2.html.

```
thresholds = "gt80 | gt0 | gt300 | gt5 | gt5";
```

For each **`vx_grib_code`** listed above one or more thresholds must be specified for use in computing discrete statistics. The thresholds are specified using the Fortran conventions of gt, ge, eq, ne, lt, le for greater than, greater than or equal to, equal to, not equal to, less than, and less than or equal to, respectively. The group of thresholds for each **`vx_grib_code`** are separated by a "`|`" character. [Note: This parameter currently must be specified regardless of whether or not discrete statistics will be computed.]

---

**`message_type = "ADPUPA";`**

The Point-Stat tool performs verification using observations for one message type at a time. The **`message_type`** variable contains a space-separated list of the message types to use for verification. By default, only surface and upper air observations are used for verification. At least one **`message_type`** must be provided. See http://www,emc.ncep.noaa.gov/mmb/data_processing/prepbufr.doc/table_1.htm for a list of the possible types.

---

**`mask_grids = "G212";`**

The **`mask_grids`** variable contains a space-separated list of pre-defined NCEP grids over which to perform the Point-Stat verification. The predefined grids are specified as "**GNNN**" where **NNN** is the three digit designation for the grid. Defining a new grid would require code changes and recompiling MET. By default, `point_stat` performs verification over the NCEP Grid number 212.

---

**`mask_polys = "";`**

The **`mask_polys`** variable contains a space-separated list of files which define lat/lon polygons to be used in specifying verification regions. Several masking polygons used by NCEP are predefined in the `data/poly` subdirectory of the MET distribution. 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. By default, no masking polygons are used.

---

**`ci_alpha = "0.05";`**

The **`ci_alpha`** variable contains a space-separated list of alpha values to be used when computing confidence intervals. The confidence interval computed is 1 minus the **`ci_alpha`** value. By default, **`ci_alpha`** is set to 0.05, indicating that the 95th percentile confidence interval should be computed. Refer to Section 4.2.3 for more information about confidence intervals and recommended values.

---

```
interp_flag[] = [ 0, 0, 0, 0, 1 ];
```

The **`interp_flag`** array contains a series of flags to indicate which interpolation methods should be employed when interpolating the gridded model data to the observation points. The flags indicate what function should be applied to the neighborhood of model points around the observation. The flags correspond to the minimum, maximum, median, un-weighted mean, and the distance-weighted mean of the neighborhood of points. Setting the flag to 1 indicates that the corresponding interpolation method should be used. By default, the distance-weighted mean is used. These methods are described in Section 4.2.1.

```
interp_width = "1 2";
```

The **`interp_width`** variable contains a space-separated list of values to be used in defining the neighborhoods over which the interpolation is performed. The neighborhood is a simply square centered on the observation point. The **`interp_width`** value specifies the width of that square. An **`interp_width`** value of 1 is interpreted as the nearest neighbor model grid point to the observation point. An **`interp_width`** of 2 corresponds to the 4 closest model grid points, while an **`interp_with`** of 3 corresponds to the 9 closest, and so on. By default, the nearest neighbor and the 4 closest grid points define the neighborhoods used.

```
interp_threshold = 1;
```

The **`interp_threshold`** variable contains 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 greater 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.

```
output_flag[] = [ 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 ];
```

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 VSDB file. Setting the flag to 0 indicates that the line type should not be generated. Setting the flag to 1 indicates that the line type should be written to the VSDB file only. Setting the flag to 2 indicates that the line type should be written to the VSDB file as well as a separate ASCII file where the data is grouped by line type. The eleven output flags correspond to the following eleven types of output line types:

- FHO for Forecast, Hit, Observation Rates

- CTC for Contingency Table Counts
- CTP for Contingency Table Proportions (= CTC/Total Count)
- CFP for Contingency Forecast Proportions (= CTC/Total Forecast Count)
- COP for Contingency Observation Proportions (= CTC/Total Observation Count)
- CTS for Contingency Table Statistics
- CNT for Continuous Statistics
- SL1L2 for Scalar L1L2 Partial Sums
- SAL1L2 for Scalar Anomaly L1L2 Partial Sums when climatological data is supplied
- VL1L2 for Vector L1L2 Partial Sums
- VAL1L2 for Vector Anomaly L1L2 Partial Sums when climatological data is supplied

Note that the first five line types are easily derived from one another. The user is free to choose which measures are most desired. All of the line types are described in more detail in Section 4.3.3.

---

```
ncep_defaults = 1;
```

The **`ncep_defaults`** variable may be set to 0 (“false”) or 1 (“true”) to indicate whether or not the NCEP conventions for GRIB codes greater than 128 will be used. By default, it it set to true.

---

### *4.3.3* `point_stat` *output*

`point_stat` produces output in VSDB and, optionally, ASCII format. The ASCII output duplicates the VSDB 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 VSDB file will be named using the following naming convention: **`point_stat_YYYYMMDDHHI_FHH.vsdb`** where **YYYYMMDDHH** indicates the model initialization time and **HH** indicates the forecast hour (i.e., lead time). The output ASCII files will be named similarly: **`point_stat_YYYYMMDDHHI_FHH_TYPE.txt`** where **TYPE** is one of **`fho`**, **`ctc`**, **`ctp`**, **`cfp`**, **`cop`**, **`cts`**, **`cnt`**, **`sl1l2`**, **`sal1l2`**, **`vl1l2`**, or **`val1l2`** to indicate the line type it contains.

The first 9 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.

| ***HEADER*** | | |
|---|---|---|
| **Column Number** | **Header Column Name** | **Description** |
| 1 | `VRS` | Version number used by NCEP (set to 01). |
| 2 | `MODEL` | User provided text string designating model name. |
| 3 | `F-HR` | Forecast hour (i.e., lead time). |
| 4 | `INIT_TIME` | Model Initialization time in YYYYMMDDHH format. |
| 5 | `OBTYPE` | Verifying observation type. |
| 6 | `VX_MASK` | Verifying masking region indicating the masking grid or polyline region applied. |
| 7 | *`Line Type and Threshold or Alpha Value`* | Varies by line type, see tables below. |
| 8 | `VAR` | Model variable being verified. |
| 9 | `LEVEL` | Vertical level at which verification performed. |

| ***FHO OUTPUT FORMAT*** | | |
|---|---|---|
| **Column Number** | **FHO Column Name** | **Description** |
| 7 | `FHO_T` | Forecast, Hit, Observation line type and threshold applied. |
| 10 | `TOTAL` | Total number of matched pairs. |
| 11 | `F_RATE` | Forecast rate. |
| 12 | `H_RATE` | Hit rate. |
| 13 | `O_RATE` | Observation rate. |
| 14 | `INTERP_MTHD` | Interpolation method applied. |
| 15 | `INTERP_PNTS` | Number of points in the interpolation neighborhood. |

| ***CTC OUTPUT FORMAT*** | | |
|---|---|---|
| **Column Number** | **CTC Column Name** | **Description** |
| 7 | `CTC_T` | Contingency Table Counts line type and threshold applied. |
| 10 | `TOTAL` | Total number of matched pairs. |
| 11 | `FY_OY` | Number of forecast yes and observation yes. |
| 12 | `FY_ON` | Number of forecast yes and observation no. |
| 13 | `FN_OY` | Number of forecast no and observation yes. |
| 14 | `FN_ON` | Number of forecast no and observation no. |
| 15 | `INTERP_MTHD` | Interpolation method applied. |
| 16 | `INTERP_PNTS` | Number of points in the interpolation neighborhood. |

| ***CTP OUTPUT FORMAT*** | | |
|---|---|---|
| **Column Number** | **CTP Column Name** | **Description** |
| 7 | CTP_T | Contingency Table Proportion line type and threshold applied. |
| 10 | TOTAL | Total number of matched pairs. |
| 11 | FY_OY_TP | Proportion of forecast yes and observation yes of the total. |
| 12 | FY_ON_TP | Proportion of forecast yes and observation no of the total. |
| 13 | FN_OY_TP | Proportion of forecast no and observation yes of the total. |
| 14 | FN_ON_TP | Proportion of forecast no and observation no of the total. |
| 15 | FY_TP | Proportion of forecast yes of the total. |
| 16 | FN_TP | Proportion of forecast no of the total. |
| 17 | OY_TP | Proportion of observation yes of the total. |
| 18 | ON_TP | Proportion of observation no of the total. |
| 19 | INTERP_MTHD | Interpolation method applied. |
| 20 | INTERP_PNTS | Number of points in the interpolation neighborhood. |

| ***CFP OUTPUT FORMAT*** | | |
|---|---|---|
| **Column Number** | **CFP Column Name** | **Description** |
| 7 | CFP_T | Contingency Forecast Proportion line type and threshold applied. |
| 10 | TOTAL | Total number of matched pairs. |
| 11 | FY_OY_FP | Proportion of forecast yes and observation yes of the number of forecast yes. |
| 12 | FY_ON_FP | Proportion of forecast yes and observation no of the number of forecast yes. |
| 13 | FN_OY_FP | Proportion of forecast no and observation yes of the number of forecast no. |
| 14 | FN_ON_FP | Proportion of forecast no and observation no of the number of forecast no. |
| 15 | FY | Number of forecast yes. |
| 16 | FN | Number of forecast no. |
| 17 | INTERP_MTHD | Interpolation method applied. |
| 18 | INTERP_PNTS | Number of points in the interpolation neighborhood. |

| ***COP OUTPUT FORMAT*** | | |
|---|---|---|
| **Column Number** | **COP Column Name** | **Description** |
| 7 | COP_T | Contingency Observation Proportion line type and threshold applied. |
| 10 | TOTAL | Total number of matched pairs. |
| 11 | FY_OY_OP | Proportion of forecast yes and observation yes of the number of observation yes. |
| 12 | FY_ON_OP | Proportion of forecast yes and observation no of the number of observation no. |
| 13 | FN_OY_OP | Proportion of forecast no and observation yes of the number of observation yes. |
| 14 | FN_ON_OP | Proportion of forecast no and observation no of the number of observation no. |
| 15 | OY | Number of observation yes. |
| 16 | ON | Number of observation no. |
| 17 | INTERP_MTHD | Interpolation method applied. |
| 18 | INTERP_PNTS | Number of points in the interpolation neighborhood. |

| ***CTS OUTPUT FORMAT*** | | |
|---|---|---|
| **Column Number** | **CTS Column Name** | **Description** |
| 7 | CTS_T | Contingency Table Statistics line type and threshold applied. |
| 10 | TOTAL | Total number of matched pairs. |
| 11 | BASER | Base rate. |
| 12 | FMEAN | Forecast mean. |
| 13 | ACC | Accuracy. |
| 14 | BIAS | Frequency Bias. |
| 15 | PODY | Probability of detecting yes. |
| 16 | PODN | Probability of detecting no. |
| 17 | POFD | Probability of false detection. |
| 18 | FAR | False alarm ratio. |
| 19 | CSI | Critical Success Index. |
| 20 | GSS | Gilbert Skill Score. |
| 21 | HK | Hanssen-Kuipers Discriminant. |
| 22 | HSS | Heidke Skill Score. |
| 23 | ODDS | Odds Ratio. |
| 24 | INTERP_MTHD | Interpolation method applied. |
| 25 | INTERP_PNTS | Number of points in the interpolation neighborhood. |

| ***CNT OUTPUT FORMAT*** | | |
|---|---|---|
| **Column Number** | **CNT Column Name** | **Description** |
| 7 | `CNT/ALPHA` | Continuous statistics line type and alpha value applied. |
| 10 | `TOTAL` | Total number of matched pairs. |
| 11-13 | `FBAR, _CL, _CU` | Forecast mean including upper and lower confidence limits. |
| 14-16 | `FSTDEV, _CL, _CU` | Standard deviation of the forecast including upper and lower confidence limits. |
| 17-19 | `OBAR, _CL, _CU` | Observation mean including upper and lower confidence limits. |
| 20-22 | `OSTDEV, _CL, _CU` | Standard deviation of the observation including upper and lower confidence limits. |
| 23-25 | `CORR, _CL, _CU` | Pearson correlation coefficient including upper and lower confidence limits. |
| 26-28 | `ME, _CL, _CU` | Mean error (F-O) including upper and lower confidence limits. |
| 29-31 | `ESTDEV, _CL, _CU` | Standard deviation of the error including upper and lower confidence limits. |
| 32 | `FBIAS` | Frequency bias. |
| 33 | `MAE` | Mean absolute error. |
| 34 | `MSE` | Mean squared error. |
| 35 | `BCMSE` | Bias-corrected mean squared error. |
| 36 | `RMSE` | Root mean squared error. |
| 37-41 | `E10, E25, E50, E75, E90` | 10th, 25th, 50th, 75th, and 90th percentiles of the error. |
| 42 | `INTERP_MTHD` | Interpolation method applied. |
| 43 | `INTERP_PNTS` | Number of points in the interpolation neighborhood. |

| ***SL1L2 OUTPUT FORMAT*** | | |
|---|---|---|
| **Column Number** | **SL1L2 Column Name** | **Description** |
| 7 | `SL1L2` | Scalar L1L2 line type. |
| 10 | `TOTAL` | Total number of matched pairs of forecast (f) and observation (o). |
| 11 | `FBAR` | Mean(f) |
| 12 | `OBAR` | Mean(o) |
| 13 | `FOBAR` | Mean(f*o) |
| 14 | `FFBAR` | Mean($f^2$) |
| 15 | `OOBAR` | Mean($o^2$) |
| 16 | `INTERP_MTHD` | Interpolation method applied. |
| 17 | `INTERP_PNTS` | Number of points in the interpolation neighborhood. |

| ***SAL1L2 OUTPUT FORMAT*** | | |
|---|---|---|
| **Column Number** | **SAL1L2 Column Name** | **Description** |
| 7 | `SAL1L2` | Scalar Anomaly L1L2 line type. |
| 10 | `TOTAL` | Total number of matched triplets of forecast (f), observation (o), and climatological value (c). |
| 11 | `FABAR` | Mean(f-c) |
| 12 | `OABAR` | Mean(o-c) |
| 13 | `FOABAR` | Mean((f-c)*(o-c)) |
| 14 | `FFABAR` | Mean($(f-c)^2$) |
| 15 | `OOABAR` | Mean($(o-c)^2$) |
| 16 | `INTERP_MTHD` | Interpolation method applied. |
| 17 | `INTERP_PNTS` | Number of points in the interpolation neighborhood. |

| ***VL1L2 OUTPUT FORMAT*** | | |
|---|---|---|
| **Column Number** | **VL1L2 Column Name** | **Description** |
| 7 | `VL1L2` | Vector L1L2 line type. |
| 10 | `TOTAL` | Total number of matched pairs of forecast winds (uf, vf) and observation winds (uo, vo). |
| 11 | `UFBAR` | Mean(uf) |
| 12 | `VFBAR` | Mean(vf) |
| 13 | `UOBAR` | Mean(uo) |
| 14 | `VOBAR` | Mean(vo) |
| 15 | `UVFOBAR` | Mean(uf*uo+vf*vo) |
| 16 | `UVFFBAR` | Mean($uf^2+vf^2$) |
| 17 | `UVOOBAR` | Mean($uo^2+vo^2$) |
| 18 | `INTERP_MTHD` | Interpolation method applied. |
| 19 | `INTERP_PNTS` | Number of points in the interpolation neighborhood. |

| ***VAL1L2 OUTPUT FILE*** | | |
|---|---|---|
| **Column Number** | **VAL1L2 Column Name** | **Description** |
| 7 | `VAL1L2` | Vector Anomaly L1L2 line type. |
| 10 | `TOTAL` | Total number of matched triplets of forecast winds (uf, vf), observation winds (uo, vo), and climatological winds (uc, vc). |
| 11 | `UFABAR` | Mean(uf-uc) |
| 12 | `VFABAR` | Mean(vf-vc) |
| 13 | `UOABAR` | Mean(uo-uc) |
| 14 | `VOABAR` | Mean(vo-vc) |
| 15 | `UVFOABAR` | Mean((uf-uc)*(uo-uc)+(vf-vc)*(vo-vc)) |
| 16 | `UVFFABAR` | Mean($(uf-uc)^2+(vf-vc)^2$) |
| 17 | `UVOOABAR` | Mean($(uo-uc)^2+(vo-vc)^2$) |
| 18 | `INTERP_MTHD` | Interpolation method applied. |
| 19 | `INTERP_PNTS` | Number of points in the interpolation neighborhood. |

# Chapter 5 – The Grid-Stat Tool

## 5.1 Introduction

The Grid-Stat tool provides verification statistics for a matched forecast and observation grid. All of the forecast gridpoints in the region of interest are matched to observation gridpoints on the same grid. All the matched gridpoints are used to comput 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 all of the forecasts and observations are on the same grid.

Scientific and statistical aspects of the Grid-Stat tool are briefly considered in this chapter, followed by practical details regarding usage and output from the tool.

## 5.2 Scientific and statistical aspects

### *5.2.1 Statistical measures*

The statistical measures produced by the Grid-Stat tool are the same as the statistics produced by the Point-Stat tool. The only difference is that each gridpoint of the forecast field is matched to its corresponding gridpoint in the verification field (i.e., no interpolation is executed by the MET). See Section 4.2.2 for information about these statistics.

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.

### *5.2.2* **100(1 - $\alpha$)%** *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.  See section 4.2.3 for more details on confidence intervals currently provided by MET.

## 5.3 Practical information

This section contains information about configuring and running the Grid-Stat tool.  The Grid-Stat tool is used to perform verification of gridded model data using gridded observations.  The input gridded model and observation datasets must be in GRIB format or in NetCDF format as the output of the Pcp-Combine tool.  In both cases, the input model and observation dataset must be on a common grid.  The gridded observation data may be actual observations such as Stage II or Stage IV data for verifying accumulated precipitation, or a model analysis field may be used as gridded observations.

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 because the input model and observation datasets must already be on a common grid. The Grid-Stat tool computes a number of continuous statistics for the forecast minus observation differences as well as discrete statistics once the data have been thresholded.

### *5.3.1* `grid_stat` *usage*

The usage statement for the Grid-Stat tool is listed below:

```
Usage: grid_stat    fcst_file obs_file  model
                    [ -config config_file ]
                    [ -outdir output_dir ]
                    [ -v level ]
```

`grid_stat` has three required arguments and up to three optional ones.

***Required arguments for*** **`grid_stat`**

1. The **fcst_file** argument identifies the GRIB file or NetCDF output of `pcp_combine` containing the model data to be verified.

2. The **obs_file** argument indicates the GRIB file or the NetCDF output of pcp_combine containing the gridded observations to be used for the verification of the model.

3. The **model** argument is a short text string indicating a name for the model being verified. The **model** text string will be written to the output VSDB and ASCII files. When data from multiple models is combined in a database, the **model** string is used to distinguish the models that were verified.

***Optional arguments for*** **`grid_stat`**

1. The **`-config`** **config_file** argument identifies the name of the configuration file to be used instead of the default configuration. The contents of the configuration file are discussed in more detail below.

2. The **`-outdir`** **output_dir** indicates the directory where output files should be written.

3. The **-v level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 1. 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.

An example of the `grid_stat` calling sequence is listed below:

Example 1:
```
grid_stat sample_fcst.grb
          sample_obs.grb
          WRF
          -config GridStatConfig
```

In Example 1, the Grid-Stat tool is instructed to 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. The name chosen for the model data is simply **WRF**.

A second example of the `grid_stat` calling sequence is listed below:

Example 2:
```
grid_stat sample_fcst.nc
          sample_obs.nc
          WRF
          -config GridStatConfig
```

In the second example, the Grid-Stat tool is instructed to 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 using 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.

### *5.3.2* `grid_stat` *configuration file*

The default configuration file for the Grid-Stat tool, named **GridStatConfig_default**, can be found in the `data/config` directory in the MET distribution. 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 `-config` option must be invoked if using a configuration file other than the default. The default configuration file contains many comments describing its contents. The contents are also described in more detail below.

```
vx_grib_code = "61/A3";
```

The `vx_grib_code` variable contains a space-separated list of model variables and corresponding vertical levels to be verified. The GRIB code itself or the corresponding abbreviation may be used to specify which model fields are to be verified. Each GRIB code must be followed by a level indicator in the form "**ANNN**", "**LNNN**", "**PNNN**", or "**PNNN-NNN**" for an accumulation interval, a single vertical level, a single pressure level, or a range of pressure levels. "**NNN**" indicates the accumulation or level value. By default, `grid_stat` will verify accumulated precipitation (GRIB code 61) with a 3-hourly accumulation interval.
A list of GRIB codes is available at
http://www.nco.ncep.noaa.gov/pmb/docs/on388/table2.html.

```
thresholds = "gt0.0 ge5.0";
```

For each `vx_grib_code` listed above one or more thresholds must be specified for use in computing discrete statistics. The thresholds are specified using the Fortran conventions of `gt`, `ge`, `eq`, `ne`, `lt`, `le` for greater than, greater than or equal to, equal to, not equal to, less than, and less than or equal to, respectively. The group of thresholds for each `vx_grib_code` are separated by a "`|`" character. By default, the field of 3-hourly accumulated precipitation will be thresholded greater than zero and greater than or equal to 5.0 mm. It is the user's responsibility to know the units for each model variable and to choose appropriate threshold values.

```
mask_grids = "G212";
```

The **mask_grids** variable contains a space-separated list of pre-defined NCEP grids over which to perform the `grid_stat` verification. The predefined grids are specified as "**GNNN**" where **NNN** is the three digit designation for the grid. Defining a new grid would require code changes and recompiling MET. By default, `grid_stat` performs verification over the NCEP Grid number 212. See Appendix B for a list of grids that will be accepted.

```
mask_polys = "";
```

The `mask_polys` variable contains a space-separated list of files which define lat/lon polygons to be used in specifying verification regions. Several masking polygons used by NCEP are predefined in the `data/poly` subdirectory of the MET distribution. 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. By default, no masking polygons are used.

```
ci_alpha = "0.05";
```

The **`ci_alpha`** variable contains a space-separated list of alpha values to be used when computing confidence intervals. The confidence interval computed is 1 minus the **ci_alpha** value. By default, **`ci_alpha`** is set to 0.05, indicating that the 95th percentile confidence interval should be computed. Refer to Section 4.2.3 for more information about confidence intervals.

---

```
output_flag[] = [ 2, 2, 2, 2, 2, 2, 2, 2, 1 ];
```

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 VSDB file except for the last one. Setting the flag to 0 indicates that the line type should not be generated. Setting the flag to 1 indicates that the line type should be written to the VSDB file only. Setting the flag to 2 indicates that the line type should be written to the VSDB file as well as a separate ASCII file where the data are grouped by line type. The first eight output flags correspond to the following eight types of output line types:

1. FHO for Forecast, Hit, Observation Rates
2. CTC for Contingency Table Counts
3. CTP for Contingency Table Proportions (= CTC/Total Count)
4. CFP for Contingency Forecast Proportions (= CTC/Total Forecast Count)
5. COP for Contingency Observation Proportions (= CTC/Total Observation Count)
6. CTS for Contingency Table Statistics
7. CNT for Continuous Statistics
8. SL1L2 for Scalar L1L2 Partial Sums

Note that line types 1-5 are easily derived from one another. The user is free to choose which measure is most desired. See Section 5.3.3 for more information about the information in the output files.

The ninth and last flag in the **`output_flag`** array indicates whether or not the raw forecast minus observation difference field should be written to a NetCDF file. Setting the flag to 1 indicates that the NetCDF file should be created, while setting it to 0 disable its creation.

---

```
ncep_defaults = 1;
```

The **`ncep_defaults`** variable may be set to 0 (“false”) or 1 (“true”) to indicate whether or not the NCEP conventions for GRIB codes greater than 128 will be used. By default, it is set to true.

---

### *5.3.3* `grid_stat` *output*

`grid_stat` produces output in VSDB and, optionally, ASCII and NetCDF formats. The ASCII output duplicates the VSDB 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 VSDB file will be named using the following naming convention: **`grid_stat_YYYYMMDDHHI_FHH.vsdb`** where **YYYYMMDDHH** indicates the model initialization time and **HH** indicates the forecast hour (i.e., lead time).

The output ASCII files will be named similarly: **`grid_stat_YYYYMMDDHHI_FHH_TYPE.txt`** where **TYPE** is one of **`fho`**, **`ctc`**, **`ctp`**, **`cfp`**, **`cop`**, **`cts`**, **`cnt`**, or **`sl1l2`** to indicate the line type it contains.

The first 9 header columns are common to all of the output files generated by the Grid-Stat tool. Please refer to the tables in section 4.3.3 (`point_stat` output) for a description of the output VSDB and optional ASCII files.

If requested in the **`output_flag`** array, a NetCDF file containing the raw forecast minus observation difference fields for each combination of variable type/level and masking region applied will be generated. The output NetCDF file will be named similar to the other output files: **`grid_stat_YYYYMMDDHHI_FHH_diff.nc`**. One may use commonly available NetCDF utilities such as `ncdump` or `ncview` to view the contents of the output file.

These NetCDF files contain the dimensions and variables shown in the following tables.

| `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). |

| `grid_stat` ***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. |
| `DIFF_VAR_LVL_MASK` | lat, lon | Foreach model variable (VAR), vertical level (LVL), and masking region (MASK), the difference (forecast – observation) is computed for each point in the mask. |

# Chapter 6 – The MODE Tool

## 6.1 Introduction

This chapter provides an outline of the Method for Object-Based Diagnostic Evaluation (MODE) tool, which was developed by the Verification Group at the Research Applications Laboratory, NCAR/Boulder, USA. More information about MODE can be found in Davis et al. (2006a,b) and Brown et al. (2007).

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 on the worldwide web at http://www.rap.ucar.edu/projects/icp/index.html.

MODE resolves objects in both the forecast and observed fields which 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 field. 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.

## 6.2 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.

### *6.2.1 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, $\phi$ is our filter function, and $C$ is the resulting convolved field. The variables $(x, y)$ and $(u, v)$ are grid coordinates. The filter function $\phi$ is a simple circular filter determined by a radius of influence $R$, and a height $H$:

$$\phi(x,y) = H \text{ if } x^2 + y^2 \le R^2 \text{, and } \phi(x,y) = 0 \text{ otherwise.}$$

The parameters $R$ and $H$ are not independent. They are related by the requirement that the integral of $\phi$ 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 $C$ field is in hand, it is thresholded to create a mask field $M$:

$$M(x,y) = 1 \text{ if } C(x,y) \ge T \text{, and } M(x,y) = 0 \text{ otherwise.}$$

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 the entire process of resolving objects in the raw data field is controlled by two parameters – the radius of influence $R$, and the threshold $T$.

An example of the steps involved on resolving objects is shown in Figure 6-1. Fig. 6-1a 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, Fig. 6-1d show the objects once the original precipitation values have been restored to the interiors of the objects.

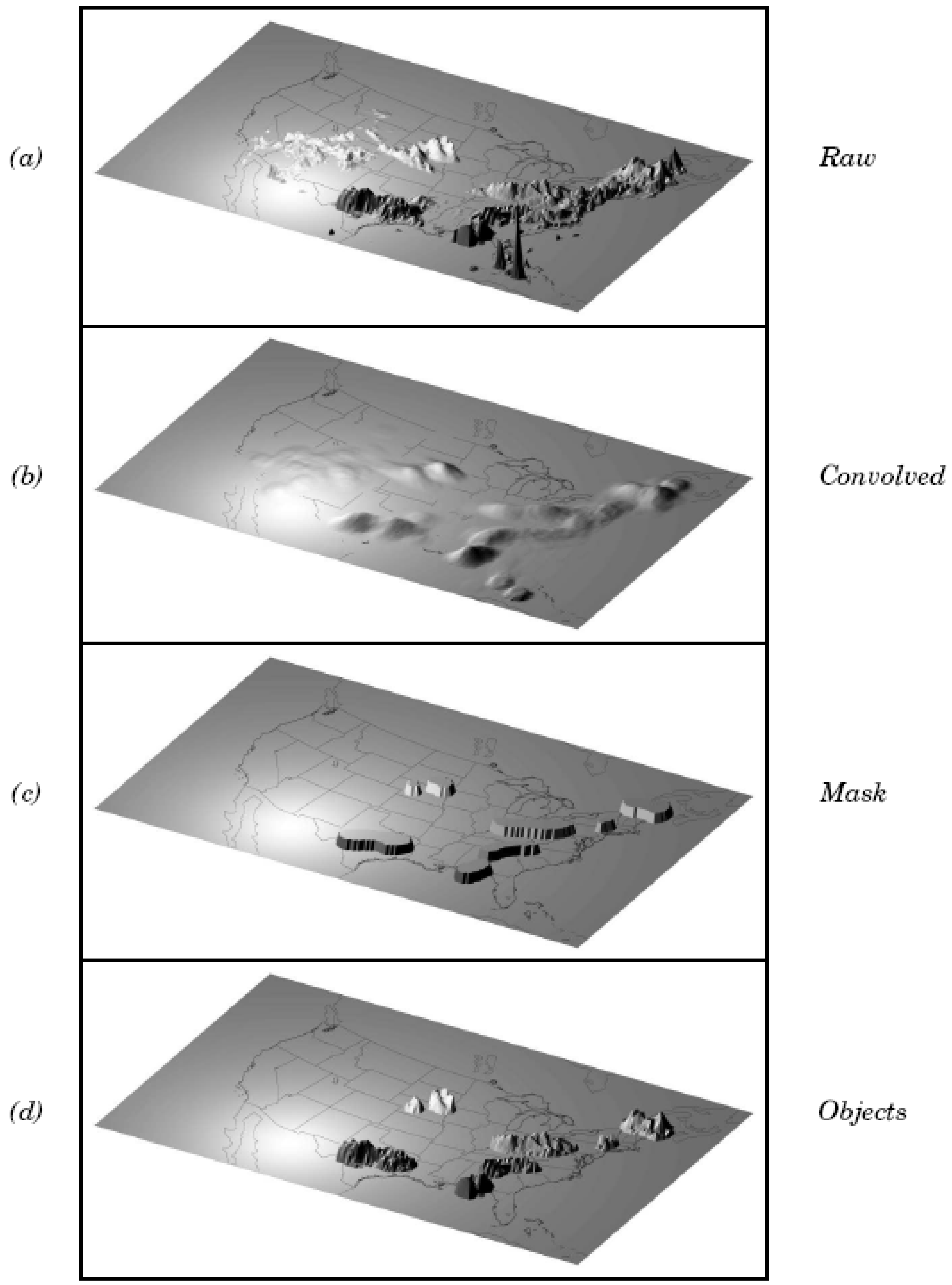


***Figure 6-1****: Example of an application of the MODE object identification process to a model precipitation field.*

### 6.2.2 Attributes

Object attributes are defined both for single objects and for object pairs. Typically 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 deemed not to be 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

$$S_x = \sum_{x,y} x\xi(x, y) \quad \text{and} \quad S_y = \sum_{x,y} y\xi(x, y) \, .$$

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 an large, extended object.

*Axis Angle*, denoted by $\theta$, is calculated from the second moments. It gives information on the orientation or "tilt" of an object.

*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 rectangle divided by its length.

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 is simply the (vector) difference between the centroids of the two objects. Another example would be *angle difference*, the difference in the axis angles.

Several area measures are also used for pair attributes. *Union Area* is the total area that's in either one (or both) of the two objects. *Intersection Area* is the area that's inside both objects simultaneously. *Symmetric Difference* is the area inside at least one object, but not inside both.

### *6.2.3 Fuzzy logic*

Once object attributes $\alpha_1, \alpha_2, ..., \alpha_n$ are determined, 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 $\alpha_i$ 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 are of little interest, and high values are of 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 (or measured) value of an attribute. The confidence maps generally are functions of the entire attribute vector $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$, in contrast to the interest maps, where each $I_i$ is a function only of $\alpha_i$. To see why this is necessary, imagine an electronic wind vane that outputs a stream of numerical values for 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. There has to be at least enough wind to overcome friction and turn the vane. Thus, in this case, our confidence in one attribute (wind direction) is dependent on the value of another attribute (wind speed). In MODE, all the confidence maps except that for axis angle are set to a constant value of 1. The axis angle confidence is a function of aspect ratio, values near one having low confidence, and values far from one having high confidence.

Next, scalar *weights* $w_i$ are assigned to each attribute, representing some judgment regarding the relative importance of the various attributes. As an example, when MODE was first being developed, centroid distance was weighted quite a bit higher than any of the other attributes, because it was felt that storm systems being located close to each other spatially was 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* $T$, 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 is then thresholded, and pairs of objects that have $T$ values above this 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 was thresholded to produce the mask field. A second (smaller) threshold can be specified so that objects that are separated at the higher threshold but joined at the lower threshold are merged.

### *6.2.4 Summary statistics*

Once MODE has been run, summary statistics are written to an output file. These files contain information about all single and composite 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. (See the examples using `awk` in Chapter 7.) Refer to Section 6.3.3 for lists of the statistics included in the `mode` output files. Examples in R and IDL will be provided in the future.

## 6.3 Practical information

This section contains a description of configuring and running the MODE tool. 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 GRIB format or in NetCDF format as the output of the pcp_combine tool. In both cases, the input model and observation dataset must be on a common grid. The gridded observation data may be actual observations such as Stage II or Stage IV data for verifying accumulated precipitation, or a model analysis field may be used as gridded observations. However, the user is cautioned that it is generally unwise to verify model output using an analysis field produced by the same model.

The MODE tool provides the capability of selecting a single model variable/level from which to derive objects to be analyzed. The MODE tool 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, the MODE tool 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, the MODE tool offers a wide range of flexibility in how the objects are defined, processed, matched, and merged.

### *6.3.1* `mode` *usage*

The usage statement for the MODE tool is listed below:

```
Usage: mode           fcst_file obs_file
                      [ -config config_file ]
                      [ -config_merge merge_config_file ]
```

```
                    [ -outdir output_dir ]
                    [ -v level ]
                    [ -plot ]
                    [ -obj_plot ]
                    [ -obj_stat ]
                    [ -ct_stat ]
```

`mode` has two required arguments and can take up to 8 optional ones.

### *Required arguments for* `mode`

1. The **fcst_file** argument indicates the GRIB file or NetCDF output of `pcp_combine` containing the model field to be verified.

2. The **obs_file** argument indicates the GRIB file or the NetCDF output of `pcp_combine` containing the gridded observations to be used for the verification of the model.

### *Optional arguments for* `mode`

1. The **`-config`** **config_file** option indicates the name of the configuration file to be used instead of the default configuration. The contents of the configuration file will be discussed shortly.

2. The **`-config_merge`** **merge_config_file** argument 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.

3. The **`-outdir`** **output_dir** indicates the directory where output files should be written.

4. The **`-v`** **level** option indicates the desired level of verbosity. The contents of "level" will override the default setting of 1. 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 **`-plot`** option disables the generation of the output PostScript plot containing a summary of the features-based verification technique.

6. The **-obj_plot** option disables the generation of the output NetCDF file containing the forecast and observation simple and composite object fields.

7. The **-obj_stat** option disables the generation of the output ASCII file containing the attributes of the simple and composite objects and pairs of objects.

8. The **-ct_stat** option disables the generation of the output ASCII file containing the contingency table counts and statistics for the raw, filtered, and object fields.

An example of the MODE calling sequence is listed below:

Example 1
```
mode            sample_fcst.grb
                sample_obs.grb
                -config WrfModeConfig
```

In Example 1, the MODE tool is instructed to 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 **WrfModeConfig** file.

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

Example 2
```
mode            sample_fcst.nc
                sample_obs.nc
                -config WrfModeConfig
```

In Example 2, the MODE tool is instructed to 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 **WrfModeConfig** file. Since the model and observation files contain only a single field of accumulated precipitation, the **WrfModeConfig** file should specify that accumulated precipitation be verified.

### *6.3.2* `mode` *configuration file*

The default configuration file for the MODE tool, named **WrfModeConfig_default**, can be found in the `data/config` directory in the MET distribution. 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. The `-config` option must be invoked when using a configuration file other than the default. Each configuration file contains many comments describing its contents. Descriptions of **WrfModeConfig_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.

```
grid_res = 4;
```

The **`grid_res`** variable is defined as the nominal spacing for each grid square in kilometers. The variable is not used directly in the code, but subsequent variables in the configuration file are defined in terms of it. Therefore, setting this appropriately will help ensure that appropriate default values are used below.

```
vx_grib_code = "APCP/A3";
```

The **`vx_grib_code`** variable is used to specify the model variable and corresponding vertical level to be verified. The GRIB code itself or the corresponding abbreviation may be used to specify which model field is to be verified. Each GRIB code must be followed by a level indicator in the form "**`A`NNN**", "**`L`NNN**", or "**`P`NNN**" for an accumulation interval, a single vertical level, or a single pressure level. "**NNN**" indicates the accumulation or level value. By default, `mode` will verify accumulated precipitation (GRIB code 61) with a 3-hourly accumulation interval. A list of GRIB codes is available at http://www.nco.ncep.noaa.gov/pmb/docs/on388/table2.html.

```
mask_missing_flag = 0;
```

The **`mask_missing_flag`** variable specifies how missing data in the raw model and observation fields will be treated. A value of 0 indicates that no additional processing is to be done. A value of 1 indicates that the missing data in the observation field should be used to mask the forecast field. A value of 2 indicates that the missing data in the forecast field should be used to mask the observation field. A value of 3 indicates that the masking should be performed in both directions. Prior to defining objects, it is recommended to make the raw fields look similar. However, by default no masking is performed.

```
mask_grid = "";
```

The **`mask_grid`** variable specifies a pre-defined NCEP grid with which to mask the raw forecast and observation fields. The predefined grids are specified as "**`G`NNN**" where **NNN** is the three digit designation for the grid. By default, no masking grid is applied. A list of these masks is presented in Appendix B.

```
mask_grid_flag = 0;
```

The **`mask_grid_flag`** variable specifies how the **`mask_grid`** should be applied.

- `0` indicates that the masking grid should not be applied.
- `1` indicates that the masking grid should be applied to the forecast field.

- `2` indicates that the masking grid should be applied to the observation field.
- `3` indicates that the masking grid should be applied to both fields.

By default, the masking grid is not applied.

```
mask_poly = "";
```

Similar to the **`mask_grid`** variable above, the **`mask_poly`** variable specifies the name of a file which defines a lat/lon polygon to be used in masking the raw forecast and observation fields. Several masking polygons used by NCEP are predefined in the `data/poly` subdirectory of the MET distribution. By default, no masking polygons are used.

```
mask_poly_flag = 0;
```

Similar to the **`mask_grid_flag`** variable above, the **`mask_poly_flag`** variable specifies how the masking polygon should be applied.

- `0` indicates the masking polygon should be applied to neither field.
- `1` indicates the masking polygon should be applied to the forecast field.
- `2` indicates the masking polygon should be applied to the observation field.
- `3` indicates that the masking polygon should be applied to both fields.

By default, the masking polygon is not applied.

```
fcst_raw_threshold   = "ge0.0";
obs_raw_threshold    = "ge0.0";
```

The **`fcst_raw_threshold`** and **`obs_raw_threshold`** variables are used to threshold the raw fields. Prior to defining objects, it is recommended to make the raw fields look similar. For example, if the model only predicts values for a variable above some threshold, the observations should be thresholded at that same level. The thresholds are specified using the Fortran conventions of `gt`, `ge`, `eq`, `ne`, `lt`, `le` for greater than, greater than or equal to, equal to, not equal to, less than, and less than or equal to, respectively. By default, the raw fields are thresholded greater than or equal to zero.

```
fcst_conv_radius      = 60/grid_res;
obs_conv_radius      = 60/grid_res;
```

The **`fcst_conv_radius`** and **`obs_conv_radius`** variables define 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`** variable.

```
bad_data_threshold   = 0.5;
```

The **bad_data_threshold** variable 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 **bad_data_threshold** is set to 0.5.

```
fcst_conv_threshold  = "ge5.0";
obs_conv_threshold  = "ge5.0";
```

The **fcst_conv_threshold** and **obs_conv_threshold** variables specify the threshold values to be applied to the convolved field to define objects. The thresholds are specified using the Fortran conventions of **gt**, **ge**, **eq**, **ne**, **lt**, **le** described previously. By default, objects are defined using a convolution threshold of 5.0.

```
fcst_area_threshold  = "ge0";
obs_area_threshold   = "ge0";
```

The **fcst_area_threshold** and **obs_area_threshold** variables specify the area threshold values to be applied to the defined objects. The area of an object is simply a count of the number of grid squares which comprise it. A user may, for example, want to only consider objects which meet some minimum size criteria. The thresholds are specified using the Fortran conventions of **gt**, **ge**, **eq**, **ne**, **lt**, **le** described previously. By default, all objects are retained since the area thresholds are set to greater than or equal to zero.

```
fcst_inten_perc                  = 100;
fcst_inten_perc_threshold = "ge0.0";
obs_inten_perc                   = 100;
obs_inten_perc_threshold = "ge0.0";
```

The **fcst_inten_perc**, **fcst_inten_perc_threshold**, **obs_inten_perc**, and **obs_inten_perc_threshold** variables specify the intensity threshold values to be applied to the defined objects. For each object defined, the intensity values within the object are sorted, and the requested intensity percentile value is computed. By default, the maximum value is computed since the intensity percentiles are set to 100. Any objects whose intensity percentile does not meet the corresponding intensity percentile threshold specified will be discarded. A user may, for example, want to only consider objects which meet some maximum intensity criteria. By default, the intensity percentile threshold applied is greater than or equal to zero.

```
fcst_merge_threshold = "ge1.25";
obs_merge_threshold  = "ge1.25";
```

The **`fcst_merge_threshold`** and **`obs_merge_threshold`** variables are used to define larger objects for use in merging the original objects. These variables define the threshold value used in the double thresholding merging technique. Note that in order to use this merging technique, it must be requested using the **`fcst_merge_flag`** and **`obs_merge_flag`**. These thresholds should be chosen to define larger objects which fully contain the originally defined objects. For example, for objects defined as **`ge5.0`**, a merge threshold of **`ge2.5`** will define larger objects which 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.

---

```
fcst_merge_flag = 1;
obs_merge_flag = 1;
```

The **`fcst_merge_flag`** and **`obs_merge_flag`** variable control what type of merging techniques will be applied to the objects defined in each field.

- `0` indicates that no merging should be applied.
- `1` indicates that the double thresholding merging technique should be applied.
- `2` indicates that objects in each field should be merged by comparing the objects to themselves using a fuzzy engine approach.
- `3` indicates that both techniques should be used.

By default, the double thresholding merging technique is applied.

---

```
match_flag = 1;
```

The **`match_flag`** variable 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`** controls what type of merging is allowed in this context.

- `0` indicates that no matching should be performed between the fields at all.
- `1` indicates that additional merging is allowed in both fields.
- `2` indicates that additional merging is allowed only in forecast field.
- `3` indicates that no additional merging is allowed, meaning that only the object pair with the highest interest value will be matched.

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`** variable is used to specify how far apart objects should be to consider 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 variable 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`** variable.

```
centroid_dist_weight            = 2.0;
boundary_dist_weight            = 4.0;
convex_hull_dist_weight         = 0.0;
angle_diff_weight               = 1.0;
area_ratio_weight               = 1.0;
int_area_ratio_weight           = 2.0;
complexity_ratio_weight         = 0.0;
intensity_ratio_weight          = 0.0;
```

The **`weight`** variables 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 object **area ratio**, the **intersection divided by the union area 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 sums are normalized by the sum of the weights listed above.

```
intensity_percentile         = 50;
```

The **`intensity_percentile`** variable corresponds to the **`intensity_ratio_weight`** variable listed above. The **`intensity_percentile`** should be set between **0** and **100** to define which percentile of intensity should be compared for pairs of objects. By default, the 50th percentile, or median value, is chosen.

```
centroid_dist_if = {
   (   0.0,           1.0 )
   (  40.0/grid_res, 1.0 )
   ( 400.0/grid_res, 0.0 )
};

boundary_dist_if = {
   (   0.0,           1.0 )
   ( 160.0/grid_res, 1.0 )
   ( 800.0/grid_res, 0.0 )
};

convex_hull_dist_if = {
   (   0.0,          1.0 )
   ( 160.0/grid_res, 1.0 )
   ( 800.0/grid_res, 0.0 )
};

angle_diff_if = {
   (  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_if(x) = ratio_if(x);

int_area_ratio_if = {
   ( 0.00, 0.00 )
   ( 0.10, 0.50 )
   ( 0.25, 1.00 )
   ( 1.00, 1.00 )
};

complexity_ratio_if(x) = ratio_if(x);

intensity_ratio_if(x) = ratio_if(x);
```

The set of interest function variables listed above are used to 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 the documentation for the configuration file language on the MET User's website (http://www.dtcenter.org/met/users) for more detail. By default, many of these functions are defined in terms of the previously defined `grid_res` variable.

```
aspect_ratio_conf(t) = ( (t - 1)^2/(t^2 + 1) )^0.3;
```

The `aspect_ratio_conf` variable defines a confidence function applied to the angle difference attribute. Objects whose aspect ratio is nearly one will be close to circular in shape. Therefore, the object's angle is not well defined. Thus, lower confidence is given to the computed angle difference attribute.

```
area_ratio_conf(t) = t;
```

The **`area_ratio_conf`** variable defines a confidence function applied to the centroid distance attribute. For two objects which are very different in size, the area ratio will be close to zero. Therefore, less confidence is given to the importance of the distance between their centroids.

```
total_interest_threshold  = 0.7;
```

The **`total_interest_threshold`** variable should be set between **0** and **1**. This threshold is applied to the total interest values computed for each pair of objects. Object pairs whose interest value is above this threshold will be matched, while those that are below this threshold will remain unmatched. Increasing the threshold will decrease the number of matches while decreasing the threshold will increase the number of matched. By default, the total interest threshold is set to 0.7.

---

```
print_interest_threshold  = 0.0;
```

The **`print_interest_threshold`** variable is used to determine which pairs of object attributes will be written to the output object attribute ASCII file. The user may choose to set the **`print_interest_threshold`** to the same value as the **`total_interest_threshold`**, meaning that only object pairs which 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`** variable.

```
zero_border_size = 4;
```

The MODE tool is not able to define objects which touch the edge of the grid. After the convolution step is performed the outer columns and rows of data are zeroed out to enable MODE to identify objects. The **`zero_border_size`** variable specifies how many outer columns and rows of data are to be zeroed out.

```
raw_color_table = "MET_BASE/data/colortables/mode_raw.ctable";
mode_color_table = "MET_BASE/data/colortables/mode_obj.ctable";
```

The **`raw_color_table`** and **`mode_color_table`** variables indicate which color table files are to be used when generating the output PostScript plot. The raw field are plotted using values in the **`raw_color_table`**, while the object identified are plotted using the values in the **`mode_color_table`**. By default, these variables point to color tables in the MET distribution. Users are free to create their own color tables following the format in the examples. Note that the range defined for the default raw color table is

0 to 1. By convention, when a color table is defined with a range of 0 to 1, it will be scaled to match the actual range of the data present in raw field.

**`ncep_defaults = 1;`**

The **`ncep_defaults`** variable may be set to **`0`** ("false") or **`1`** ("true") to indicate whether or not the NCEP conventions for GRIB codes greater than 128 will be used. By default, it is set to true.

### *6.3.3* `mode` *output*

MODE produces output in ASCII, NetCDF, and PostScript formats.

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. The second 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. This file is named using the naming convention: **mode_VAR_LVL_fcst_obs_YYYYMMDDHHI_FHH_HHA_cts.txt** where **VAR** is the variable being verified, **LVL** indicates the vertical level, **YYYYMMDDHHI** indicates the model initialization time, **FHH** indicates the model forecast hour (i.e., lead time), and **HHA** indicates the accumulation period. The **`cts`** string stands for contingency table statistics. The generation of this file can be disabled using the **`-ct_stat`** command line option. Please refer to the CTS table in section 4.3.3 (`point_stat` output) for a description of the contents of a CTS file.

The second ASCII file the MODE tool generates contains all of the attributes for the single objects, the merged composite 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 number of lines in this file depends on the number of objects defined. This file contains lines of 6 types which are indicated by the contents of the "**object_id**" column. The "**object_id**" can take the following 6 forms: **FNNN**, **ONNN**, **FNNN_ONNN**, **CFNNN**, **CONNN**, **CFNNN_CONNN**. In each case, **NNN** is a three digit number indicating the object index. The first two forms correspond to attributes for simple forecast and simple observation objects. These rows contain valid data in columns 1-32 and fill data elsewhere. The next form corresponds to attributes for pairs of simple forecast and observation objects. These rows contains valid data in columns 1-12 and 33-44 and fill data elsewhere. The next two forms correspond to attributes for merged composite forecast and observation objects. These rows contains valid data in columns 1-32 and fill data elsewhere. And the last form corresponds to matched pairs of composite forecast and observation objects. These rows contain valid data in columns 1-12 and 33-43 and fill data elsewhere. The generation of this file can be disabled using the **`-obj_stat`** command line option.

***A note on terminology***: a composite object need not necessarily consist of more than one simple object. A composite 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 composite objects as well.

The contents of the columns in this ASCII file are summarized in the table below.

| `mode` ***ASCII OBJECT ATTRIBUTE OUTPUT FORMAT*** | | |
|---|---|---|
| **Column Number** | **MODE Column Name** | **Description** |
| 1 | `F-HR` | Forecast hour (i.e., lead time). |
| 2 | `ACCUM-HR` | Hours of accumulation (for precipitation). |
| 3 | `INIT_TIME` | Model initialization time in YYYYMMDDHH format. |
| 4 | `FCST_RAD` | Forecast convolution radius in grid squares. |
| 5 | `FCST_THR` | Forecast convolution threshold. |
| 6 | `OBS_RAD` | Observation convolution radius in grid squares. |
| 7 | `OBS_THR` | Observation convolution threshold. |
| 8 | `VAR` | Model variable being verified. |
| 9 | `LEVEL` | Vertical level at which verification performed. |
| 10 | `OBJECT_ID` | Object numbered from 1 to the number of objects in each field. |
| 11 | `OBJECT_CAT` | Object category indicating to which composite object it belongs. |
| 12 | `=` | Delimiter. |
| 13-14 | `CENTROID_X, _Y` | Location of the centroid in grid units. |
| 15-16 | `CENTROID_LAT, _LON` | Location of the centroid in lat/lon. |
| 17 | `AXIS_ANG` | Axis angle of the object. |
| 18 | `LENGTH` | Length of the enclosing rectangle in grid units. |
| 19 | `WIDTH` | Width of the enclosing rectangle in grid units. |
| 20 | `AREA` | Area as a count of grid squares. |
| 21 | `AREA_FILTER` | Number of grid squares within the object containing non-zero data in the filtered field. |
| 22 | `CURVATURE` | Curvature of the object defined in terms of third order moments. |
| 23-24 | `CURVATURE_X, _Y` | Center of curvature in grid coordinates. |
| 25 | `COMPLEXITY` | Complexity of the object defined by comparing the area of an object to the area of its convex hull. |
| 26-30 | `INTENSITY_10, _25, _50, _75, _90` | 10th, 25th, 50th, 75th, and 90th percentiles of intensity of the filtered field within the object. |
| 31 | `INTENSITY_NN` | The percentile of intensity chosen for use in the |

| mode *ASCII OBJECT ATTRIBUTE OUTPUT FORMAT* | | |
|---|---|---|
| **Column Number** | **MODE Column Name** | **Description** |
| | | percentile intensity ratio (column 43). |
| 32 | `INTENSITY_SUM` | Sum of the intensities of the filtered field within the object. |
| 33 | `CENTROID_DIST` | Distance between two objects centroids in grid units. |
| 34 | `BOUNDARY_DIST` | Minimum distance between the boundaries of two objects in grid units. |
| 35 | `CONVEX_HULL_DIST` | Minimum distance between the convex hulls of two objects in grid units. |
| 36 | `ANGLE_DIFF` | Difference between the axis angles of two objects. |
| 37 | `AREA_RATIO` | Ratio of the areas of two objects defined as the minimum of the forecast area divided by the observation area and its reciprocal |
| 38 | `INTERSECTION_AREA` | Intersection area of two objects as a count of grid squares. |
| 39 | `UNION_AREA` | Union area of two objects as a count of grid squares. |
| 40 | `SYMMETRIC_DIFF` | Symmetric difference of two objects as a count of grid squares. |
| 41 | `INTESECTION_OVER_UNION_AREA` | Intersection area divided by union area. |
| 42 | `COMPLEXITY_RATIO` | Ratio of complexities of two objects defined as the minimum of the forecast complexity divided by the observation complexity and its reciprocal. |
| 43 | `PERCENTILE_INTENSITY_RATIO` | Ratio of the nth percentile (column 31) of intensity of the two objects defined as the minimum of the forecast intensity divided by the observation intensity and its reciprocal. |
| 44 | `INTEREST` | Total interest value computed for a pair of simple objects. |

The MODE tool creates a NetCDF output file containing the object fields defined. The NetCDF file contains 4 gridded fields: indices for the simple forecast objects, indices for the simple observation objects, indices for the matched composite forecast objects, and indices for the matched composite observation objects. The NetCDF file also contains lat/lon data for each grid point so that NetCDF utilities can accurately display the data. The generation of this file can be disabled using the **`-obj_plot`** command line option.

The dimensions and variables included in the `mode` NetCDF files are described in the following tables.

| **`mode` *NetCDF OUTPUT FILE 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). |

| **`mode` *NetCDF OUTPUT FILE 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. |
| `fcst_obj_id` | `lat, lon` | Simple forecast object id number for each grid point. |
| `fcst_comp_id` | `lat, lon` | Composite forecast object id number for each grid point. |
| `obs_obj_id` | `lat, lon` | Simple observation object id number for each grid point. |
| `obs_comp_id` | `lat, lon` | Composite observation object id number for each grid point. |

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 **`-plot`** command line option.

The PostScript plot will contain four 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.

The first page contains a great deal of summary information. Six tiles of images display thumbnail versions of the raw fields, matched/merged object fields, and object index fields for the forecast and observation. 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 royal blue indicates unmatched objects. 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 plot display enlargements of the forecast and observation raw and object fields, respectively. The fourth page of the plot, displays the forecast object with the outlines of the observation objects overlaid, and vice versa.

If the double threshold merging technique or the fuzzy engine merging technique is applied, those steps will be summarized on additional pages.

# Chapter 7 – Scripting

## 7.1 Example scripts for running MET tools

This section provides examples of the use of the Bourne shell sh as a scripting language to run some of the MET tools. The example scripts allow a MET user to process many files at once.

Suppose we want to run the Pcp-Combine tool on a collection of precipitation files in the directory `/home/user/my_pcp_dir`. We want a precipitation accumulation period of 12 hours, and a valid accumulation period of 24 hours. We'll suppose the data are from August 2006. We can do this using the following script:

**Script 7-1**

```
 1  #!/bin/sh
 2
 3  pcp_accum_period=12
 4  valid_accum_period=24
 5  day=1
 6
 7  while [ "$day" -le 31 ]
 8  do
 9     if [ "$day" -le 9 ]
10     then
11        ds=0$day
12     else
13        ds=$day
14     fi
15     pcp_combine \
16          0000-00-00_00:00:00 \
17          $pcp_accum_period \
18          2006-08-$ds.00:00:00 \
19          $valid_accum_period \
20          2006-08-$ds.pcp.nc \
21          -pcp_dir /home/user/my_pcp_dir
22     day=$((day + 1))
23  done
```

The first line in Script 7-1 tells the operating system to use `/bin/sh` to execute the commands in the file. If the Bourne shell resides somewhere else on your system, you'll have to change this accordingly. On lines 3 and 4 some variables are defined to hold the precip accumulation hours and the valid accumulation hours. Line 5 initializes the day variable to 1. This variable will be the day of the month.

Line 7 starts the loop over the days in the month. The loop ends on line 23. Since `pcp_combine` needs dates and times formatted in a certain way, the variable `ds` (for daystring) is created, which contains the day of the month with a leading "0" if the day is less than 10. This happens in lines 9–14.

The Pcp_Combine tool is run in lines 15–21. We've split the command over several lines so that the script would fit on this page. Line 16 is the `pcp_init_time` argument. We've set it to all zeroes so that `pcp_combine` will look at all files in the directory. Line 17 is the precipitation accumulation period, defined in line 3. Line 18 is the valid time. Here we use the `ds` variable that we calculated in lines 9–14. Line 19 contains the valid accumulation period variable, defined in line 4. Line 20 is our output file name. In this example, the output file names include the calendar date and a suffix of "`.pcp.nc`".

Line 21 tells `pcp_combine` where to look for input files. Finally, in line 22, the last line in the body of the loop increments the `day` variable.

In the second example, we'll use the grid stat tool to process some August 2006 data. Suppose our obs files are in `/d1/user/obs` and the forecast files are in `/d1/user/fcst`. We want the VSDB files written to `/d1/user/gridstat_out`.

**Script 7-2**

```
1   #!/bin/sh
2
3   model=wrf22
4   config_file=/home/user/my_grid_stat_config
5   obs_dir=/d1/user/obs
6   fcst_dir=/d1/user/fcst
7   day=1
8
9   while [ "$day" -le 31 ]
10  do
11     if [ "$day" -le 9 ]
12     then
13        ds=0$day
14     else
15        ds=$day
16     fi
17     grid_stat \
18          $fcst_dir/2006-08-$ds.fcst.pcp.nc \
19          $obs_dir/2006-08-$ds.obs.pcp.nc \
20          $model \
21          -outdir /d1/user/gridstat_out \
22          -config config_file
23     day=$((day + 1))
24  done
```

In lines 3–6 of Script 7-2, we define some useful variables: the model, the configuration file, and the directories for the observation and forecast files. As in Script 7-1, we loop through the days in the month. Line 7 initializes day to 1.

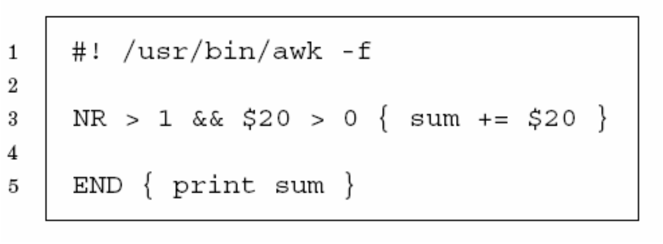
Lines 11–16 are copied from the Script 7-1. In lines 17–22 we run the Grid-Stat tool. For simplicity, we assume a simple input file naming convention that consists of the calendar data plus an appropriate suffix. Forecast and observation files are named on lines 18 and 19, respectively. Note that we use the variable `ds` here. Line 21 names our desired output directory, and line 22 tells `grid_stat` which configuration file to use. As in Script 7-1, the last line in the loop body (here, line 23) increments `day`.

## 7.2 Example scripts for use with MODE output files

Since the output of MODE is plain ASCII text in a flat tabular format, we'll use `awk` as our example scripting language. We'll concentrate on scripts to process a single MODE output file. Multiple input files can be handled by (for example) concatenating them all together and piping the result to the script. We'll show an example of that later.

As our first example, suppose you wanted to calculate the total area of the objects in the file. The following `awk` script will do that:

**Script 7-3**



```
1   #! /usr/bin/awk -f
2
3   NR > 1 && $20 > 0 { sum += $20 }
4
5   END { print sum }
```

Line numbers from the script file are indicated at the left. They are *not* part of the script. Lines 2 and 4 are blank just to enhance readability. The first line just tells the shell to use `awk` to execute the commands in the script file. If this line were missing, the shell itself would try to execute the commands. If `awk` resides somewhere other than `/usr/bin` on your system, you'll need to change this.

The logic behind line 3 is as follows: if the record number is greater than one and field #20 in the record is positive, then add the value of that field to the sum. The default record separator in `awk` is a newline character, so each record is one line in the file. We skip the first line because that's a header line. The default field separator is whitespace, so each field is one column in the line. Column #20 in the MODE output files is Area, which we add to the sum if it's positive. The flag value used in the MODE output files is −9999, so we test for positivity so that these flag values won't contribute to the sum. In `awk`, user-defined variables like `sum` begin life with a value of zero, so we don't need to initialize them.

Finally, in line 5, the action specified by `END` is executed after all records have been read. In this example, the action is to print the value of `sum`.

Script 7-4 provides a somewhat more complex example. Suppose we want to find the average centroid location for all simple forecast objects that have a 90th percentile intensity of at least 10. In the MODE output files, the ($x$, $y$) grid coordinates of the centroid are in columns 13 and 14. 90th percentile intensity is column 30, and object ID is column 11. Here’s the script:

**Script 7-4**

```
1   #! /usr/bin/awk -f
2
3   $10 ~ /^F...$/ && $30 >= 10.0 {
4      xsum += $13
5      ysum += $14
6      ++count
7   }
8
9   END {
10     xsum /= count
11     ysum /= count
12     print "Average Centroid is (" xsum ", " ysum ")"
13  }
```

For an example test MODE output file (not shown), running this script yields this output:

```
Average Centroid is (361.705, 183.135)
```

Before going through this script, we should note a slight bug. If no objects meet our criteria, then `count` will be zero, and so in lines 10 and 11 we’ll be dividing by zero. We’ll leave it to the reader to fix this.

The test in line 3 is for a character string in column 10 that consists of an “`F`” followed by three other characters, and a value in column 30 that is at least 10. The character “`^`” anchors the `F` to the beginning of the column and the dollar sign “`$`” anchors to the end of the column. If in line 3 we had instead written `$10 ~ /F.../` then we would have matched `F001` (which we want), but we would also have matched `CF002` and `CF002 CO002` (which we don’t).

Notice that we’re not explicitly skipping the header line, like we did in the first script. That isn’t needed here because the header line will fail the first of the two tests on line 3 (and probably the second one, too). That means this script would be suitable for processing more than one file. If no filename arguments are given to the script, `awk` will read from the standard input, so just `cat` the files all together and pipe the output to the script.

In general, many files can be processed without having to deal with the header lines in each file, via a simple trick involving `grep`. For example, if you have a collection of MODE output files in the directory `mydir`, you can process them all using this command:

```
cat mydir/* | grep -v INIT | myscript
```

The header line in each file contains the string "`INIT`" in one of the columns, and that string shouldn't appear in any other line in the file. So `grep -v INIT` can be used to remove header lines without losing any real data lines.

# Chapter 8 – Analysis Tools

The MET Analysis Tool (MAT) ties together results from the Point-Stat, Grid-Stat, and MODE tools by providing summary statistical information as well as spatial averaging information and a small number of graphics. The initial MET Analysis Tool will be available in the fall 2007 MET release.

## 8.1 Input

The MAT requires output from the MET Point-Stat, Grid-Stat, and MODE tools. See Sections 4.3.3, 5.3.3, and 6.3.3, respectively, for information on the output format of the Point-Stat, Grid-Stat, and MODE tools.

## 8.2 Methods

The MAT will aggregate results over a user-specified time range or spatial subset. It will also stratify statistics based on time of day, model initialization time, physics package, or output filename. Future functionality may include information about time-trends and/or calculations based on climatology (e.g., anomaly correlation).

## 8.3 Output

In addition to providing statistical summaries, the MAT will also have basic graphical capabilities. Some graphics produced may include:

- Boxplots
- Discrimination plots
- Reliability diagram
- Scatter/density plots
- Conditional quantile plots
- Color-fill/contour maps of statistics
- Height series
- Histograms

These graphical capabilities will give the user an easy way to obtain overviews of the statistics produced by the Point-Stat, Grid-Stat, and MODE tools. They can also be used as a point of comparison for your own graphics packages. As a start, a couple of scripts will be provided that use alternative graphing packages (R and IDL), some of which are described in Chapter 10, Plotting and Graphics Support.

# Chapter 9 – Plotting and Graphics Support

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. Any computing language could be used for this stage; some scripts will be provided on the MET users web page (http://www.dtcenter.org/met/users/) in IDL, R, and NCL programming languages as examples to assist users.

## 9.1 Grid-Stat examples

The plots in Figure 9-1 show a time series of Frequency Bias and Gilbert Skill Score (GSS) calculated by the Grid-Stat tool and plotted using an IDL script. The script simply reads the columnar text output from the Grid-Stat output and summarizes the results. These particular plots are based on the occurrence (or non-occurrence) of precipitation greater than 1 mm over 3 h. They show skill scores for four different configurations of model runs using different physics packages and numerics. Stage II radar-gauge estimates are used as verification observations for this exercise. Over this month-long period, the models all appear to do relatively well for the period 24 July to 28 July, as the GSS rises above 0.2 and the bias drops to near 1. The time scale and ordinate axes can be easily manipulated for closer inspection of model differences.

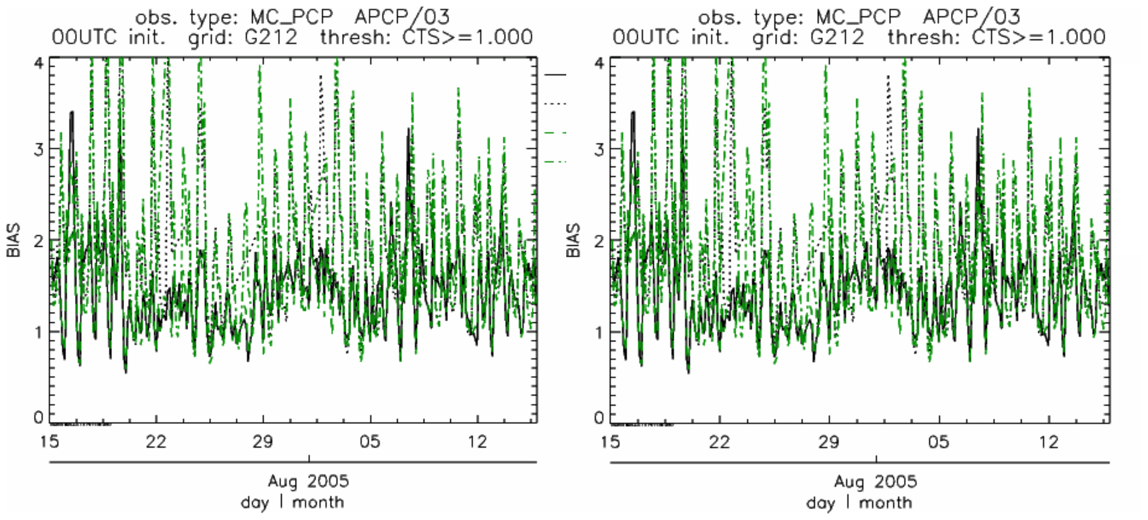


***Figure 9-1**: Time series of forecast bias and Gilbert Skill Score for several numerical models (differentiated by line-type and color) over a 32-day period in July/Aug 2005. Scores are based on the forecast of 1 mm or greater precipitation at 3-h intervals. Models were run for 24 h and were initiated every day at 00 UTC.*

A similar plot is shown in Fig. 9-2, except the data have been 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.

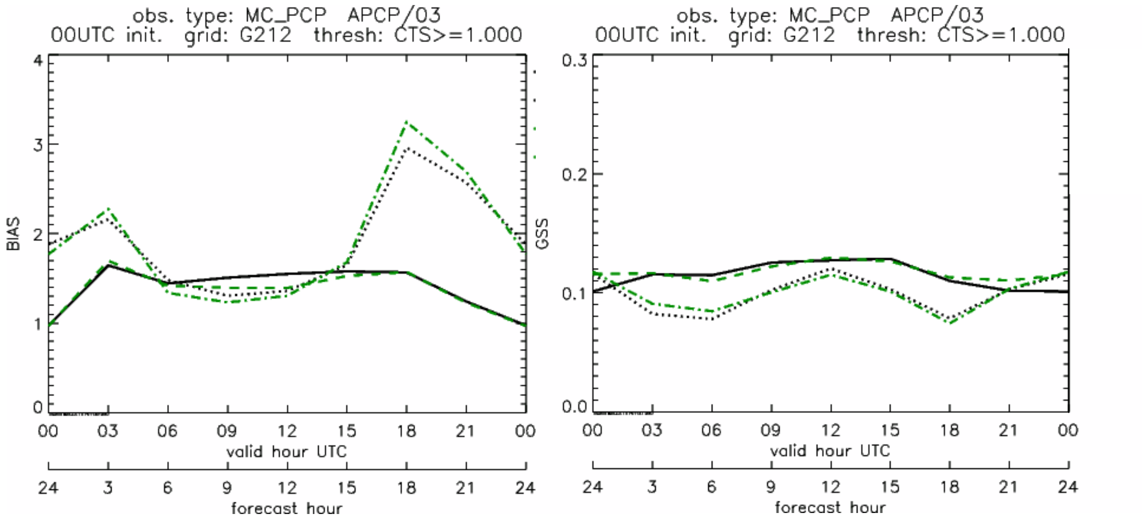


***Figure 9-2****: Time series of forecast area bias and Gilbert Skill Score for four model configurations (different lines) stratified by time-of-day. The data used to create these figures were the same as used for Fig. 9-1.*

## 9.2 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.

Fig 9-3 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 Figs. 9-1 and 9-2. 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 observed Stage II radar-gage precipitation. Note that for the 3-hr lead time, indicated as the 0300 UTC valid time in Fig. 9-3, 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.

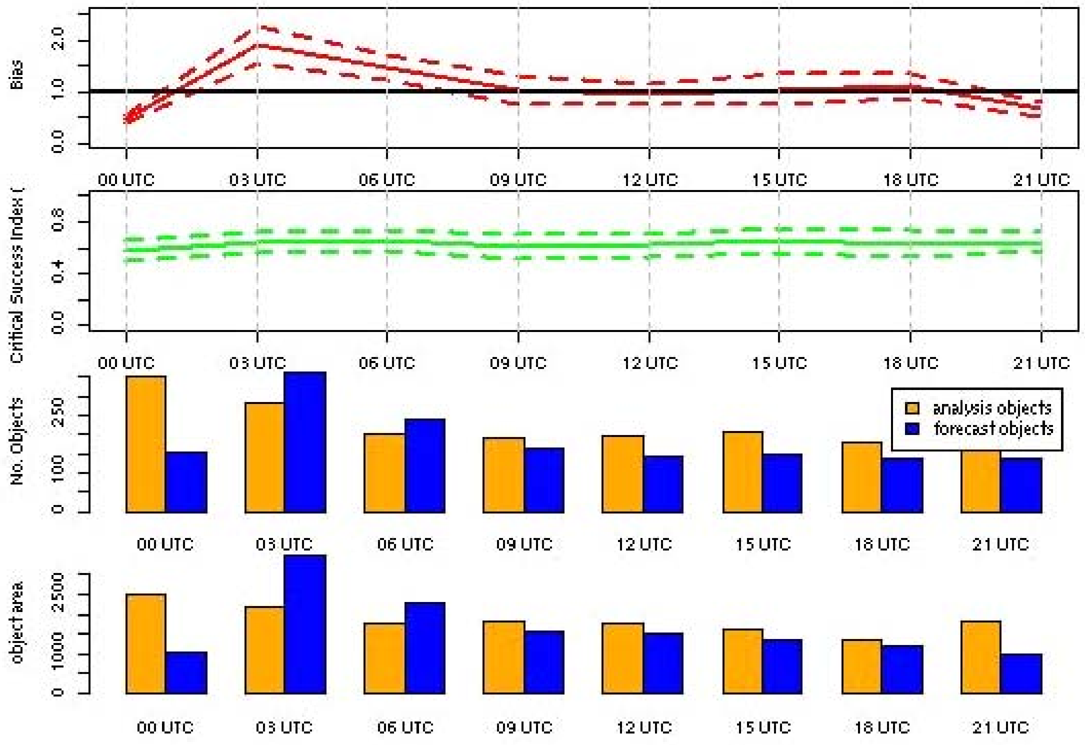


***Figure 9-3**: Traditional verification scores applied to output of the MODE tool, computed by defining matched observed objects to be hits, unmatched observed objects to be misses, and unmatched forecast objects to be false alarms; weighted by object area. Bar plots show numbers (pennultimate row) 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. Fig. 9-4, 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 Fig. 9-4 (top diagram) about 25% of the time the forecast object centroids are west of the observed object centroids, whereas in Fig. 9-4 (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, Fig 9-4 (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.

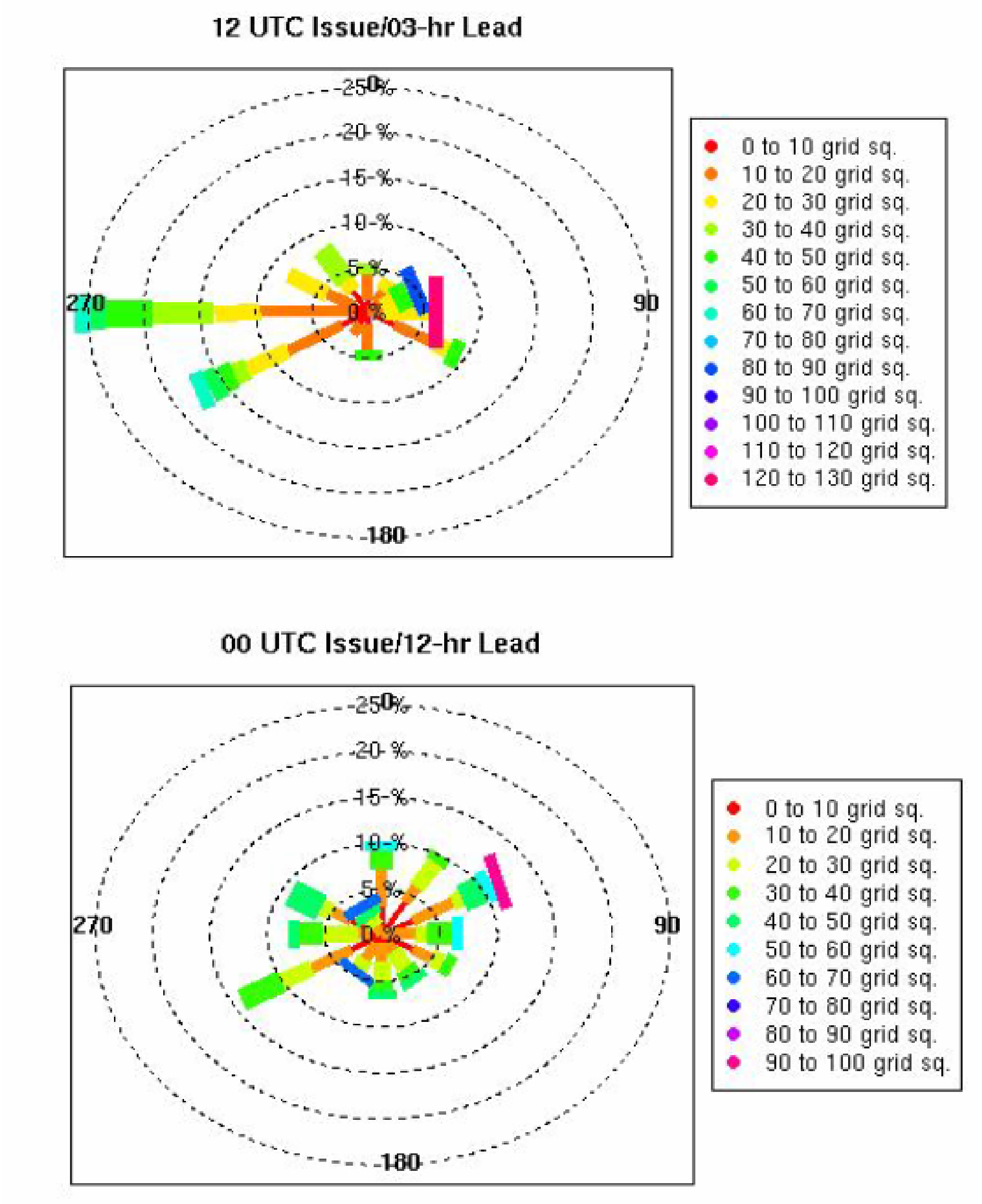


***Figure 9-4***: *Circle histograms showing object centroid angles and distances (see text for explanation).*

# References

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.

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.

Jolliffe, I.T., and D.B. Stephenson, 2003: *Forecast Verification. A Practitioner's Guide in Atmospheric Science*. Wiley and Sons Ltd, 240 pp.

Murphy, A.H., and R.L. Winkler, 1987: A general framework for forecast verification. *Monthly Weather Review*, **115**, 1330-1338.

Stephenson, D.B., 2000: Use of the “Odds Ratio” for diagnosing forecast skill. *Weather and Forecasting*, **15**, 221-232.

Wilks, D., 2006: *Statistical Methods in the Atmospheric Sciences*. Elsevier, San Diego.

# Appendix A – How do I … ?

## A.1 Frequently Asked Questions

**Q. Why was the MET written largely in C++ instead of FORTRAN?**
**A.** The 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 are VSDB and prepbufr used?**
**A.** Mainly because the first goal was to initially replicate the capabilities of other existing verification packages and make these capabilities available to both the DTC and the public. VSDB was selected as one of the output types supported.

**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.** Not yet. We plan to add support for forecast output in GRIB2 format in the next version release.

**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 plan to 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 http://www.dtcenter.org/met/users. If that doesn't answer your question, then ***email:*** ***met_help@rap.ucar.edu.***

**Q. Where are the graphics?**
**A.** Currently, there are no graphics included in this beta version. Graphics support will be added in the formal release in the fall.

## A.2 Troubleshooting

In this initial beta release, it is hard to anticipate many of the places where users may potentially have problems. Usage statements for the individual commands (`grid_stat`, `point_stat`, `mode`, `pcp_combine`, and `pb2nc`) are available by simply typing the name of the executable in the 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 the MET.

***MET won't compile***

- Are you using the correct version of the Makefile (Makefile for using the GNU C++ and F77 compilers; Makefile_PGI for the PGI compiler)
- Are the correct paths specified in the Makefile for BUFRLIB, the NetCDF and GNU Developer's Science libraries? Have these libraries been compiled and installed?
- Is the Developer's version of the GNU Science library installed? (This differs from the regular GNU Science library.)

***Grid_stat won't run***

- Are both the observational and forecast datasets on the same grid?

***MODE won't run***

- 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?

***pb2nc won't run***

- Has your prepbufr file been unblocked using NCO's cwordsh utility?

***point_stat won't run***

- Is NCO's BUFRLIB installed?
- Have you run pb2nc?

## 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)
- Lat/Lon Projection
- Exponential Projection

The following map projections will be supported in future releases of MET:

- Mercator Projection
- Polar Stereographic Projection (Southern)

## B.2 Grids

All of NCEP's pre-defined grids which 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

NCEP's grids which are pre-defined in MET are listed below by projection type:

- Lambert Conformal Projection
    - G145, G146, G163, G206, G209, G211, G212, G215, G218, G221
    - G222, G226, G227, G236, G237, G241, G245, G246, G247, G252
- Polar Stereographic Projection (Northern)
    - G005, G006, G027, G028, G055, G056, G087, G088, G100, G101
    - G103, G104, G105, G106, G107, G201, G202, G203, G205, G207
    - G213, G214, G216, G217, G223, G224, G240, G242, G249
- Lat/Lon Projection
    - G002, G003, G004, G029, G030, G033, G034, G045, G085, G086
    - G110, G175, G228, G229, G230, G231, G232, G233, G234, G243
    - G248, G250,G251

## B.3 Polylines

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 data/poly directory of the distribution. 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 polyline 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 – Supplemental Plotting Code (to be added in August)

This appendix will include a description of the supplemental code to be included with the MET in future releases.