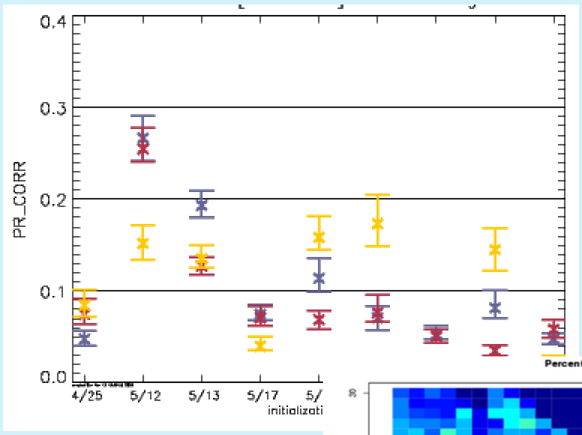
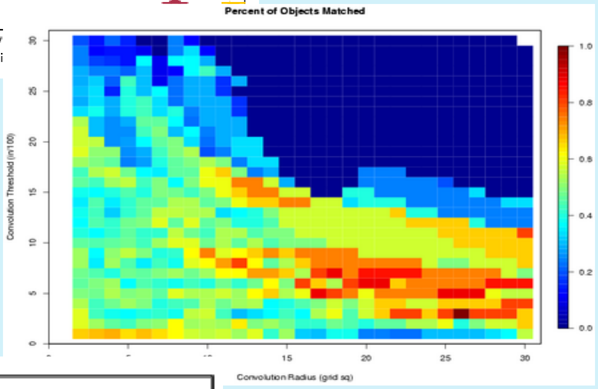
WEATHER RESEARCH & FORECASTING

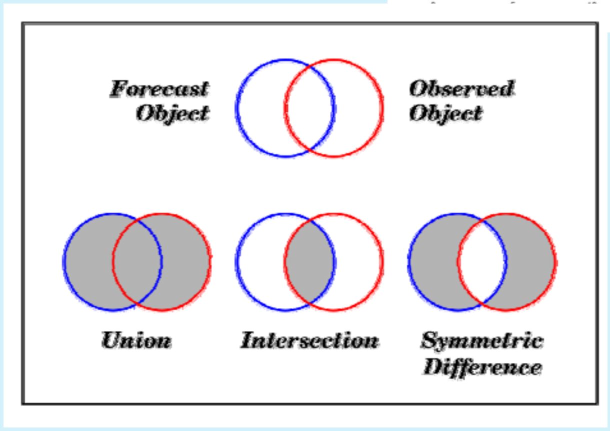
MET

Version 1.1 Model Evaluation Tools User's Guide

July 2008







Developmental Testbed Center

Model Evaluation Tools Version 1.1 (METv1.1)

User's Guide

Developmental Testbed Center

Boulder, Colorado, USA

July 2008

Contents

Section **Page**

Foreword: A note to MET users

This user's guide is provided as an aid to users of the Model Evaluation Tools (MET). MET is a set of verification tools developed by the Developmental Testbed Center (DTC) for use by the numerical weather prediction community – 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 first official release of MET (METv1.0) was on January 7, 2008. Future releases are expected in Winter 2009, and will include new capabilities and enhancements as well as corrections to any errors or system issues. Intermediate releases may include bug fixes. In the future, MET will also 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 MET planning workshops in February 2007 and April 2008; the ideas generated at those workshops will help MET grow in future years. Finally, we would like to specifically thank Elizabeth Ebert and Tressa Fowler for their guidance and other contributions. The DTC is sponsored by the National Oceanic and Atmospheric Administration (NOAA), AFWA, and the National Science Foundation (NSF). NCAR is sponsored by the National Science Foundation (NSF).

Chapter 1 – Overview of MET

1.1. Purpose and organization of the User's Guide

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

The User's Guide is organized as follows. Chapter 1 provides an overview of MET and its components. Chapter 2 contains basic information about how to get started with MET – including system requirements; required software (and how to obtain it); how to download MET; and information about compilers, libraries, and how to build the code. Chapter 3 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. Chapters 7 and 8 focus on the analysis modules, VSDB-Analysis and MODE-Analysis, which aggregate the output statistics from the other tools across multiple cases. Finally, Chapters 9 and 10 include some additional tools and information for scripting MET runs and plotting MET results. The appendices provide further useful information, including answers to some typical questions (Appendix A: How do I…?); and links and information about map projections, grids, and polylines (Appendix B). Appendices C and D provide more information about the verification measures and confidence intervals that are provided by MET. Sample code that can be used to perform analyses on the output of MET and create particular types of plots of verification results will be made available on the MET website (http://www.dtcenter.org/met/users/). Note that the MET development group also accepts contributed code which may be posted on the MET website for use by the community.

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

1.2 The Developmental Testbed Center (DTC)

MET has been developed, and will be maintained and enhanced, by the Developmental Testbed Center (DTC; http://www.dtcenter.org/). The main goal of the DTC is to serve as a bridge between operations and research, to facilitate the activities of these two important components of the numerical weather prediction (NWP) community. The DTC provides an environment that is functionally equivalent to the operational environment in which the research community can test model enhancements; the operational community benefits from DTC testing and evaluation of models before new models are implemented operationally. MET 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.

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

1.3 MET goals and design philosophy

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

The MET package has been designed to be modular and adaptable. For example, individual modules can be applied without running the entire set of tools. New tools can easily be added to the MET package due to this modular design. In addition, the tools can readily be incorporated into a larger "system" that may include a database as well as more sophisticated input/output and user interfaces. Currently, the MET package is a set of tools that can easily be applied by any user on their own computer platform.

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, governments, the private sector, and operational modeling and prediction centers.

1.4 MET components

The major components of the MET package are represented in Figure 1-1. The main stages represented are input, reformatting, intermediate output, statistical analyses, and output and aggregation/analysis. Each of these stages is described 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.

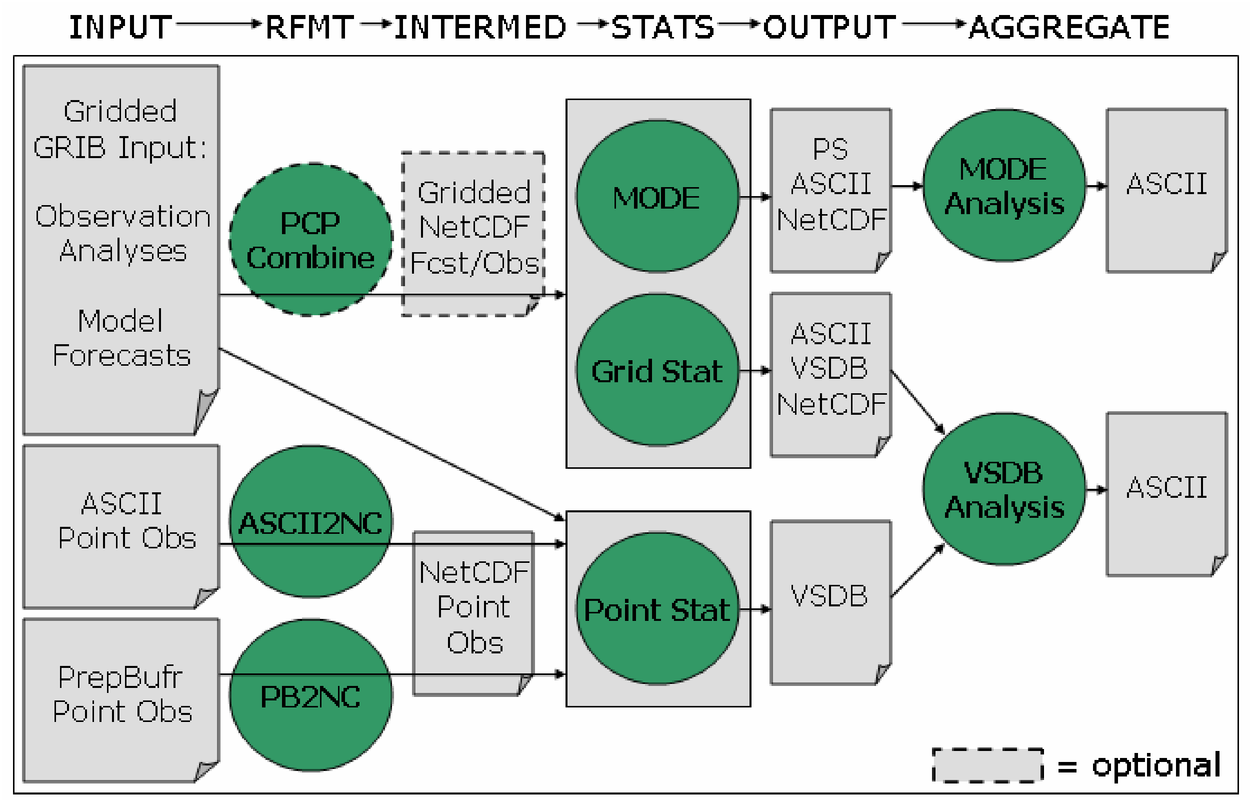


Figure 1-1. Basic representation of current MET structure and modules. Green areas represent software and modules included in MET and gray areas represent input and output files..

The reformatting stage of MET consists of the PB2NC, ASCII2NC, and Pcp-Combine tools. The PB2NC tool is used to create NetCDF files from input prepbufr files containing point observations. Likewise, the ASCII2NC tool is used to create NetCDF files from input ASCII point observations. These NetCDF files are then used in the statistical analysis step. 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 forecast field against a point-based observation (i.e., surface observing stations, ACARS, rawinsondes, and other observation types that could be described as a point observation). In addition to providing traditional forecast verification scores for both continuous and categorical variables, confidence intervals are also produced using parametric and non-parametric methods. Confidence intervals take into account the uncertainty associated with verification statistics due to sampling variability and limitations in sample size. 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. The Grid-Stat tool also now includes new "neighborhood" spatial methods, such as the Fractional Skill Score (Roberts and Lean 2008). These methods are discussed in Ebert (2008).

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

Results from the statistical analysis stage are output in ASCII, NetCDF and Postscript formats. The Point-Stat and Grid-Stat tools create output in a VSDB-like (Verification System DataBase) format. VSDB, which was developed by the National Centers for Environmental Prediction (NCEP), is a specialized ASCII format that can be easily read and used by graphics and analysis software.

Note that the VSDB-like output format of the Point-Stat and Grid-Stat tools is not exactly the same as the format developed by NCEP. Additional columns of data have been added to the end of the VSDB lines to store interpolation information that is not included in the NCEP version. Also, additional VSDB line types have been added to store statistics not included in the NCEP version. The time formats in the NCEP version have been expanded from HH to HHMMSS and from YYYYMMDDHH to

YYYYHHDD_HHMMSS to handle data at intervals finer than hourly. Throughout this document, the term "VSDB" is used to refer to the VSDB-like format described above.

The VSDB-Analysis and MODE-Analysis tools aggregate the output statistics from the previous steps across multiple cases. The VSDB-Analysis tool reads the VSDB output of Point-Stat and/or Grid-Stat and can be used to filter the VSDB data and produce aggregate continuous and categorical statistics. The MODE-Analysis tool reads the ASCII output of the MODE tool and can be used to produce summary information about object location, size, and intensity (as well as other object characteristics) across one or multiple cases.

1.5 Future development plans

MET is an evolving verification software package. New capabilities are planned in controlled, successive version releases. Bug fixes and user-identified problems will be addressed as they are found and posted to the known issues section of the MET Users web page (www.dtcenter.org/met/users/support). Plans are also 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
- Hurricane track verification
- Wind direction verification

Support for other input formats

- Support for gridded data in GRIB2
- Support for gridded data in NetCDF, CF convention

Additional analysis capabilities and plotting routines

- Post to the MET website sample analysis and plotting routines written in IDL, R, GrADS, NCL, or other languages that may include
  - Boxplots
  - Discrimination plots
  - Reliability diagrams
  - Scatter/density plots
  - Color-fill/contour maps of statistics
  - Height series
  - Histograms

Other capabilities

- Analysis tools for model comparisons

- Graphical user interface (GUI)
- Autoconf configurability

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 may include a database for model and observation storage and input/output.

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/).

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

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

Chapter 2 – Software Installation/Getting Started

2.1 Introduction

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

2.2 Supported architectures

The MET package was developed on Debian Linux using the GNU compilers and the Portland Group (PGI) compilers. The MET package has also been built on several other Linux distributions using either the GNU or PGI compilers. The MET package has also been ported to IBM machines using the IBM 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 and others	GNU g++/g77 or gfortran
DELL	XEON	Debian Linux and others	PGI pgCC/pgf77
IBM	SP Power-3/4	AIX	IBM xlC/xlf90

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. Users and contributors may develop entirely new tools 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 `g++/g77` compilers and the Portland Group (PGI) pgCC/pgf77 compilers. The MET package has also been ported to IBM machines using the IBM xlC/xlf90 compilers. As additional compilers are successfully tested, they will be added to the list of supported platforms/compilers.

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

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

In order to control the desired flow through MET, users are encouraged to run the tools via a script (see Chapter 9 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 and optional utilities

Four external libraries are required for compiling/building MET and should be downloaded and installed before attempting to install 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 that should be from the same family of compilers used when building MET.

2. Several tools within MET use Unidata's **NetCDF** libraries for writing output NetCDF files. NetCDF libraries are distributed and supported by Unidata and are freely available for download from Unidata's website at http://www.unidata.ucar.edu/software/netcdf. The same family of compilers used to build NetCDF should be used when building MET. MET is compatible with most NetCDF version 3 releases, but it is not compatible with NetCDF version 4.

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

4. The **F2C (or G2C) Library** is used by MET to enable the PB2NC tool, written in C++ to communicate with the BUFRLIB, written in Fortran. If F2C (or G2C) is not already installed on your system, it may be downloaded from the Netlib website at http://www.netlib.org/f2c. Download the file "libf2c.zip" and refer to the README file for installation instructions.

Three additional utilities are strongly recommended for use with MET:

1. The **WRF Post-Processor** is recommended for 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 re-gridding model and observation datasets in GRIB format to a common verification grid. This utility is highly recommended when using either **`grid_stat`** or **`mode`**. 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. The **cwordsh** utility, available from NCEP, is used to Fortran-block PrepBufr files prior to processing them through MET. The cwordsh utility is supported by NCEP and is freely available for download from NCEP's website (http://www.nco.ncep.noaa.gov/sib/decoders/BUFRLIB/toc/cwordsh). The input PrepBufr files must be Fortran-blocked in order to be opened as 'UNFORMATTED' files and read by NCEP's BUFRLIB. Follow the link above for more details.

2.6 Installation of required libraries

As described in section 2.5, 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/g77 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 *.c`**
   - **`g77 -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 *.c`**
   - **`pgf77 -c -DUNDERSCORE -Mnosecond_underscore *.f`**
   - **`ar crv libbufr.a *.o`**

   Compiling the BUFRLIB using the IBM xlc/xlf90 compilers consists of the following 3 steps:

   - **`xlc -c -DUNDERSCORE *.c`**
   - **`xlf -c -qextname *.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. Detailed installation instructions are available from Unidata at http://www.unidata.ucar.edu/software/netcdf/docs/netcdf-install/

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

MET requires access to the GSL source headers and library archive file at build time.

2.7 Installation of 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 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. Installation instructions for the WRF Post-Processor can be found in Chapter 2 of the WRF-NMM User's Guide or online at http://www.dtcenter.org/wrf-nmm/users/docs/user_guide/WPS/ .

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

3. NCEP's **`cwordsh`** utility performs 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 MET's **pb2nc** utility.

2.8 MET directory structure

Once you have downloaded the MET tarball and unzipped and unpacked its contents, the top-level MET directory structure follows this outline:

- **METv1.1/**
    - **Makefile_gnu**
    - **Makefile_pgi**
    - **Makefile_ibm**
    - **README**
    - **bin/**
    - **data/**
        - **colortables/**
        - **config/**
        - **map/**
        - **poly/**
        - **ps/**
        - **sample_fcst/**
        - **sample_obs/**
    - **doc/**
        - **MET_Users_Guide.pdf**
    - **lib/**
    - **out/**
    - **scripts/**
        - **config/**
    - **src/**
        - **ascii2nc/**
        - **grid_stat/**
        - **mode/**
        - **mode_analysis/**
        - **pb2nc/**
        - **pcp_combine/**
        - **point_stat/**
        - **vsdb_analysis/**

The top-level MET directory consists of a README file, three 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 will be modified in Section 2.9.

When MET has been successfully built, the `bin/` directory will contain executables for each module of MET (**`grid_stat`**, **`mode`**, **`mode_analysis`**, **`pb2nc`**, **`ascii2nc`**, **`pcp_combine`**, **`point_stat,`** and **`vsdb_analysis`**)

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/` subdirectory contains predefined lat/lon polyline regions for use in selecting regions over which to verify. The polylines defined correspond to verification regions used by NCEP as described in Appendix B. The `config/` subdirectory contains default 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 seven 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 the `config/` subdirectory. The output from the test scripts in this directory will be written to the `out/` directory. Users are encouraged to copy sample configuration files to another location and modify them for their own use.

2.9 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 Section 2.6 on how to install the required libraries.

2. **Configure the top-level Makefile**
   - Once you have downloaded the MET tarball, unzip and unpack its contents (refer to Section 2.8).
   - Make a copy of the Makefile most similar to your OS and compiler. For example, if compiling on Linux using the GNU compilers:
     - **`cp Makefile_gnu Makefile`**
   - Edit the top-level **`Makefile`** as follows:
     - Set **`MAKE`** to the full path for the GNU Make utility.
     - Set **`CXX`** to the full path for your C++ compiler.
     - Set **`FC`** to the full path for your Fortran compiler.
     - Set **`NETCDF_BASE`** to the location where NetCDF is installed if it is not installed in a standard location. The NetCDF directory should contain `include/` and `lib/` subdirectories.

- Set **`BUFR_BASE`** to the location where BUFRLIB is installed if it is not installed in a standard location.
- Set **`GSL_BASE`** to the location where the GNU Scientific Library is installed if it is not installed in a standard location. The GSL directory should contain `include/gsl/` and `lib/` subdirectories.
- Set **`F2C_BASE`** to the location where the F2C or G2C library is installed if it is not installed in a standard location.
- Set **`F2C_LIBNAME`** to either **`-lf2c`** or **`-lg2c`** to indicate which library is to be used.
- The additional parameters in the Makefile may be set as needed to configure the build to your system such as compiler flags and additional libraries.

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.

2.10 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 a test Bourne shell script for each of the eight tools in MET. However, the **`test_all.sh`** script will run the other eight 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 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 warnings or errors occur, please refer to Appendix A on troubleshooting for common problems.
    - In particular, the script that runs the PB2NC tool may error out. If this occurs, you should try reblocking the sample PrepBufr files (in the `data/sample_obs/prepbufr/` subdirectory) for your machine using the `cwordsh` utility. Refer to section 3.5 for instructions on running the `cwordsh` utility.

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

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

Both the input and output file formats are described in this chapter. Sections 3.1 and 3.2 are primarily concerned with re-formatting input files into the intermediate files required by some MET modules. These steps are represented by the first three columns in the MET flowchart depicted in Fig. 1-1. Output data formats and the software modules used to reformat the data are described in later sections.

3.1 Input data formats

The MET package can handle gridded input data in GRIB version 1 format (i.e., the same as the output format produced by the WRF Post-Processor). Point observation files may be supplied in either PrepBufr or ASCII format. 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 evenly-spaced levels in the vertical.

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 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 point-based 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. The ASCII2NC tool may be used to convert ASCII point observations that are not available in the PrepBufr files into this NetCDF format for use by the Point-Stat verification tool.

3.2 Intermediate data formats

MET uses NetCDF as an intermediate file format. The Pcp-Combine, PB2NC, and ASCII2NC tools write intermediate files in NetCDF format.

The Pcp-Combine tool operates in 3 different modes. It may be used to **sum** accumulated precipitation from several GRIB files into a single NetCDF file containing the desired accumulation period. It may also be used to **add** or **subtract** the accumalted precipitation in two GRIB files directly. The command line arguments for the Ppc-Combine tool vary depending on the mode in which it is run.

The user may choose to: (1) combine the model accumulations to match the observation accumulation period, (2) combine the observation accumulations to match the model accumulation period, or (3) combine 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, then `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.

The ASCII2NC tool simply reformats ASCII point observations into the NetCDF format needed by the Point-Stat tool. The output NetCDF file from the ASCII2NC tool has a format that is identical to the format of the output from the PB2NC tool.

3.3 Output data formats

The MET package currently produces output in four basic file formats: VSDB files, ASCII files, 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 a specialized ASCII format containing one record on each line. However, a single VSDB file may contain multiple line types. The columns before the equal sign delimiter remain the same for each line type. However, the columns after the equal sign delimiter change for each line type. VSDB files can be difficult for a human to read as the quantities represented for many columns of data change from line to line.

For this reason, ASCII output is also available as an alternative for 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 in a VSDB format. 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 tools generate gridded NetCDF output. The MODE tool creates a NetCDF file containing four gridded fields for the objects identified in the forecast and observation, simple and composite object fields. The Grid-Stat tool creates a NetCDF file containing the matched forecast/observation pairs and 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 (see Chapter 2 for more on Fortran-blocking) and one configuration file.
   - **Output**: One NetCDF file containing the observations that have been retained.

2. **ASCII2NC Tool**
   - **Input**: One ASCII point observation file that has been formatted as expected.
   - **Output**: One NetCDF file containing the reformatted observations.

3. **Pcp-Combine Tool**
    - **Input**: Two or more gridded model or observation files in GRIB1 format containing accumulated precipitation to be combined to create a new accumulation interval.
    - **Output**: One NetCDF file containing the summed accumulation interval.

4. **Point-Stat Tool**
    - **Input**: One model file in GRIB1 format, at least one point observation file in NetCDF format (as the output of the PB2NC or ASCII2NC 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.

5. **Grid-Stat Tool**
    - **Input**: One model file and one observation file either in GRIB1 format or in the NetCDF format output from 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 matched pair data and difference field for each verification region and variable type/level being verified.

6. **MODE Tool**
    - **Input**: One model file and one observation file either in GRIB1 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 object indices for the gridded simple and composite object fields, and one PostScript plot containing a summary of the features-based verification performed.

7. **VSDB-Analysis Tool**
    - **Input**: One or more VSDB files output from the Point-Stat and/or Grid-Stat tools and, optionally, one configuration file containing specifications for the analysis job(s) to be run on the VSDB data.
    - **Output**: ASCII output of the analysis jobs will be printed to the screen unless redirected to a file using the “-out” option.

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

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. Depending on the machine on which MET is being run, it may be necessary to redo the Fortran blocking step on the test data. When running the PB2NC tool with other PrepBufr datasets, Fortran-blocking will also 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 name of the file 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_file
        [-pbfile prepbufr_file]
        [-valid_beg time]
        [-valid_end time]
        [-nmsg n]
        [-dump path]
        [-v level]
```

`pb2nc` has three required arguments and can take up to six 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.

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

Optional arguments for **`pb2nc`**

1. The **`–pbfile`** **prepbufr_file** option may be used to pass additional PrepBufr files to the PB2NC tool.

2. The **`–valid_begin`** **time** option in YYYYMMDD[_HH[MMSS]] format sets the beginning of the retention time window.

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

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

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

6. The **`-v`** **level** option indicates the desired level of verbosity. The value 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 `pb2nc` calling sequence is shown below:

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

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

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 available in `scripts/config`. It is recommended that users make a copy of these files prior to modifying their contents. Each configuration file contains many comments describing its contents.

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

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

- In a C-Shell: **`setenv MSG_TYP ' "ADPUPA", "ADPSFC" '`**
- In the configuration file: **`message_type[] = [ ${MSG_TYP} ];`**

The example script for running MODE included in section 9.2 provides another example of using environment variables in configuration files.

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 “message_type” refers to the type of observation from which the observation value (or “report”) was derived. The user may specify a comma-separated list of message types to be retained. Providing an empty list indicates that all message types should be retained.

```
station_id[] = [];
```

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

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

Each PrepBufr file has an observation time associated with it. Every PrepBufr message within the file has a time-offset defined relative to that file's 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. **beg_ds** indicates how many seconds relative to the file's observation time to begin retaining observations to be used for verification (the negative sign indicates this window begins **prior** to the time assigned to the PrepBufr file). **end_ds** indicates how many seconds after the file's time to stop retaining observations for verification. The time window shown above 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 (in meters) of the observations to be retained. The range shown above is set to -1000 to 100000 meters, which essentially retains every observation.

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

The **pb_report_type**, **in_report_type**, and **instrument_type** variables are used to specify comma-separated lists of PrepBufr report types, input report types, and instrument types to be retained, respectively. If left empty, all PrepBufr report types, input report types, and instrument types will be retained.

```
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 range shown above is 1 to 255, which is the current maximum possible level.

```
obs_grib_code[] = ["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 or derived. The GRIB code itself or the corresponding abbreviation may be used to specify which observation variables are to be retained or derived. The following GRIB codes may be derived: DPT, WIND, RH, MIXR, and PRMSL for dewpoint, wind speed, relative humidity, mixing ratio, and pressure reduced to MSL. The list of GRIB codes shown above indicates that specific humidity, temperature, height, and the u and v components of the wind are to be retained.

```
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. The value shown above indicates that only observations with quality marks less than or equal to 2 will be retained.

```
event_stack_flag = 1;
```

A PrepBufr message may contain duplicate observations with different quality mark values. The **event_stack_flag** indicates whether to use the observations at the top of the event stack (observation values have had more quality control processing applied) or the bottom of the event stack (observation values have had no quality control processing applied). The flag value of 1 listed above indicates the observations with the most amount of quality control processing should be used.

`level_category[] = [];`

The **`level_category`** variable is used to specify a comma-separated list of Prepbufr data level categories to retain. An empty string indicates that all level categories should be retained. Accepted values and their meanings are described in the table below. These represent the same categories available from http://www.emc.ncep.noaa.gov/mmb/data_processing/prepbufr.doc/table_1.htm

Table 3-1. *Values for the level_category option.*

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

`version = “V1.1”;`

The **`version`** indicates the version of the `pb2nc` configuration file used. Future versions of MET may include changes to `pb2nc` and the `pb2nc` configuration file. This value should not be modified.

3.5.3 PB2NC output

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

Table 3-2. *NetCDF file dimensions for pb2nc output.*

`pb2nc` ***NetCDF DIMENSIONS***	
NetCDF Dimension	**Description**
`mxstr`	Maximum string length (15)
`hdr_arr_len`	Number of entries in each PrepBufr message header array (3)
`obs_arr_len`	Number of entries in each PrepBufr observation array (5)
`nobs`	Number of PrepBufr observations in the file (UNLIMITED)
`nhdr`	Number of PrepBufr messages in the file (variable)

Table 3-3. *NetCDF variables in pb2nc output.*

`pb2nc` ***NetCDF VARIABLES***		
NetCDF Variable	**Dimension**	**Description**
`obs_arr`	nobs, obs_arr_len	Array of floats containing values for each observation including: • Reference to the entry in the hdr_arr with which this observation is associated • GRIB code corresponding to this observation type • Pressure level in hPa or accumulation interval • Height in meters above sea level • Observation value
`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 YYYYMMDD_HHMMSS format
`hdr_arr`	nhdr, hdr_arr_len	Array of floats containing values for each PrepBufr message including: • Latitude in degrees north • Longitude in degrees east • Elevation in meters above sea level

3.6 ASCII2NC tool

This section describes how to run the ASCII2NC tool. The ASCII2NC tool is used to reformat ASCII point observations into the NetCDF format expected by the Point-Stat tool. For those users wishing to verify against point observations that are not available in PrepBufr format, the ASCII2NC tool provides a way of incorporating those observations into MET. Since the ASCII2NC tool simply performs a reformatting step, no configuration file is needed.

The initial version of the ASCII2NC tool supports a single input ASCII point observation format consisting of 10 columns of data for each observation value. The ASCII2NC tool may be enhanced in future realeases of MET to support additional ASCII point observation formats directly, based on community input and resource availability.

The input ASCII point observation format consists of one row of data per observation value. Each row of data consists of 10 columns as shown in the following table.

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

3.6.1 `ascii2nc` *usage*

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

```
Usage: ascii2nc
        ascii_file
        netcdf_file
        [-format ASCII_format]
        [-v level]
```

`ascii2nc` has two required arguments and can take up to two optional ones.

Required arguments for `pb2nc`

1. The **ascii_file** argument indicates the name of the ASCII point observation file to be processed.

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

***Optional arguments for* `ascii2nc`**

1. The **`–format ASCII_format`** will be used in future releases of MET to define the ASCII point observation format contained in the ASCII point observation file. Since the ASCII2NC tool currently only reads one point observation format, users will not need to specify this argument.

2. The **`-v`** **level** option indicates the desired level of verbosity. The value 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 `ascii2nc` calling sequence is shown below:

```
Ascii2nc  sample_ascii_obs.txt
          sample_ascii_obs.nc
```

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

3.7 Pcp-Combine tool

This section contains a description of running the Pcp-Combine tool. The Pcp-Combine tool is used (if needed) to modify the precipitation accumulation intervals from two or more GRIB files into a single NetCDF file containing the desired accumulation interval, for input to the MODE and Grid-Stat tools. 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 re-gridding GRIB files. In addition, the Pcp-Combine tool will only operate on files with the same initialization time unless it is indicated to ignore the initialization time.

3.7.1 `pcp_combine` *usage*

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

```
Usage: pcp_combine
        [[-sum] sum_args] | [-add add_args] | [-subtract
subtract_args]
        [-gc code]
        [-v level]
```

The arguments to `pcp_combine` vary depending on the mode in which it is run.

Listed below are the arguments for the **sum** command:

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

Listed below are the aguements for the **add** or **subtract** commands:

```
ADD_ARGS and SUBTRACT_ARGS:
         in_file1
         accum1
         in_file2
         accum2
         out_file
```

Required arguments for the **`pcp_combine`**

1. The Pcp-Combine tool must be run with exactly one of the **`-sum`**, **`-add`**, or **`-subtract`** command line arguments with the corresponding additional arguments.

Optional arguments for **`pcp_combine`**

1. The **`-gc` code** option may be used to override the default GRIB code value of 61 – for accumulated precipitation.
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.

Required arguments for the **`pcp_combine`** ***sum command***

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

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

3. The **valid_time** argument, in YYYYMMDD_HHMMSS format, indicates the desired valid time to which the accumulated precipitation is to be summed.

4. The **out_accum** 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` ***sum command***

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

2. The **`-pcprx`** **reg_exp** option indicates the regular expression to be used in matching files in the precipitation directory specified. The contents of "**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.

Required arguments for the `pcp_combine` ***add or subtract commands***

1. The **in_file1** argument indicates the first GRIB file to be processed.
2. The **in_accum1** argument, provided in HH format, indicates the accumulation interval to be extracted from the first GRIB file.
3. The **in_file2** argument indicates the second GRIB file to be processed.
4. The **in_accum2** argument, provided in HH format, indicates the accumulation interval to be extracted from the second GRIB file. For the **add** command, these two accumulations will be added together. For the **subtract** command, the second accumulation will be subtracted from the first.

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

Example 1:
```
pcp_combine    -sum
               20050807_000000 3
               20050808_000000 24
               sample_fcst.nc
               -pcpdir ../data/sample_fcst/2005080700
```

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

Example 2:

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

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

Table 3-4. *NetCDF file dimensions for pcp_combine output.*

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

Table 3-5. *NetCDF variables for pcp_combine output.*

`pcp_combine` ***NetCDF variables***		
NetCDF variable	**Dimension**	**Description**
`lat`	`lat, lon`	Latitude value for each point in the grid
`lon`	`lat, lon`	Longitude value for each point in the grid
`GRIB Code Abbreviation`	`lat, lon`	Amount of precipitation for each point in the grid. The name of the variable matches the GRIB code abbreviation for the field.

Chapter 4 – The Point-Stat Tool

4.1 Introduction

The Point-Stat tool provides verification statistics for forecasts at observation points (as opposed to over gridded analyses). The Point-Stat tool matches gridded forecasts to point observation locations, 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 the verification statistics.

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

4.2 Scientific and statistical aspects

The statistical methods and measures computed by the Point-Stat tool are described briefly in this section. In addition, Section 4.2.1 discusses the various interpolation options available for matching the forecast gridpoint values to the observation points. The statistical measures computed by the Point-Stat tool are described briefly in Section 4.2.2 and in more detail in Appendix C. Section 4.2.3 describes the methods for computing confidence intervals that are applied to some of the measures computed by the Point-Stat tool; more detail on confidence intervals is provided in Appendix D.

4.2.1 Interpolation/matching methods

This section provides information about the various methods available in MET to match gridded model output to point observations. In these descriptions (as shown in Fig. 4-1), *P* denotes the point where the interpolated value is calculated. An interpolation width, W, is also specified. For example, a width of 2 defines a 2 x 2 square of grid points enclosing *P*, or simply the 4 grid points closest to *P*. A width of 3 defines a 3 x 3 square consistening of 9 grid points centered on the grid point closest to *P*. The example in Fig. 4-1 shows a case with W=5. Any vertical interpolation that is needed is done in natural log of pressure coordinates, except for specific humidity, which is interpolated using the natural log of specific humidity in natural log of pressure coordinates. This section describes the different options for interpolation in the horizontal.

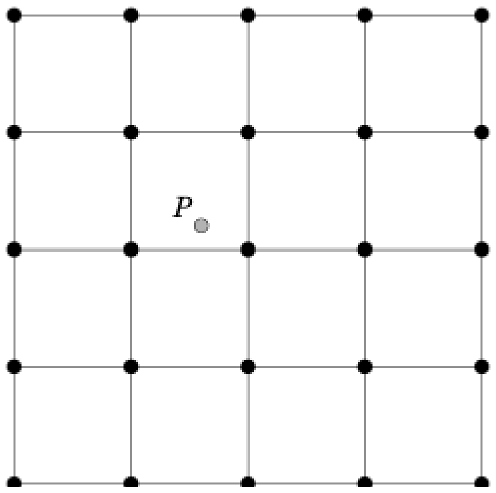


Figure 4-1: Diagram illustrating matching and interpolation methods vsed in MET. See text for explanation.

Nearest Neighbor

No interpolation is performed. The data value at *P* is simply the value at the nearest grid point. Here, "nearest" means spatially closest 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 is 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.

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.

Least-Squares Fit

To perform least squares interpolation of a gridded field at a location P, MET uses an $W \times W$ subgrid centered (as closely as possible) at P. Figure 4-1 shows the case where N = 5.

If we denote the horizontal coordinate in this subgrid by x, and vertical coordinate by y, then we can assign coordinates to the point P relative to this subgrid. These (x, y)

coordinates are chosen so that the center of the grid is $(x, y) = (0,0)$. In the figure, for example, P has coordinates $(-0.4,\ 0.2)$. Since the grid is centered near P, the coordinates of P should always be at most 0.5 in absolute value. At each of the W^2 vertices of the grid (indicated by black dots in the figure), we have data values. We would like to use these values to interpolate a value at P. We do this using least squares. If we denote the interpolated value by z, then we fit an expression of the form

$$z = \alpha x + \beta y + \gamma$$

over the subgrid. The values of α, β, and γ are calculated from the data values at the vertices. Finally, the coordinates (x, y) of P are substituted into this expression to give z, our least squares interpolated data value at P.

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 categories of measures are briefly described here; specific descriptions of the measures is provided in Appendix C. Additional information 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 partial sums and other FHO statistics that are produced by the NCEP verification system. These statistics are also described in Appendix C.

Measures for categorical variables
Categorical verification statistics are used to evaluate forecasts that are in the form of a discrete set of categories rather than on a continuous scale. Currently, Point-Stat computes categorical 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 < 273.15°K). See Appendix C for more information.

Measures for continuous variables
For continuous variables, many verification measures are based on the forecast error (i.e., $f - o$). However, it also is of interest to investigate characteristics of the forecasts, and the observations, as well as their relationship. These concepts are consistent with the general framework for verification outlined by Murphy and Winkler (1987). The statistics produced by MET for continuous forecasts represent this philosophy of verification, which focuses on a variety of aspects of performance rather than a single measure. See Appendix C for specific information.

4.2.3 Statistical confidence intervals

A single summary score gives an indication of the forecast performance, but it is a single realization from a random process that neglects uncertainty in the score's estimate. That is, it is possible to obtain a good score, but it may be that the "good" score was achieved by chance and does not reflect the "true" score. Therefore, when interpreting results from a verification analysis, it is imperative to analyze the uncertainty in the realized scores. One good way to do this is to utilize confidence intervals. A confidence interval indicates that if the process were repeated many times, say 100, then the true score would fall within the interval $100(1-\alpha)\%$ of the time. Typical values of α are 0.01, 0.05, and 0.10. The Point-Stat tool allows the user to select one or more specific α-values to use.

For continuous fields (e.g., temperature), it is possible to estimate confidence intervals for some measures of forecast performance based on the assumption that the data, or their errors, are normally distributed. The Point-Stat tool computes confidence intervals for the following summary measures: forecast mean and standard deviation, observation mean and standard deviation, correlation, mean error, and the standard deviation of the error. In the case of the respective means, the central limit theorem suggests that the means are normally distributed, and this assumption leads to the usual $100(1-\alpha)\%$ confidence intervals for the mean. For the standard deviations of each field, one must be careful to check that the field of interest is normally distributed, as this assumption is necessary for the interpretation of the resulting confidence intervals.

For the measures relating the two fields (i.e., mean error, correlation and standard deviation of the errors), confidence intervals are based on either the joint distributions of the two fields (e.g., with correlation) or on a function of the two fields. For the correlation, the underlying assumption is that the two fields follow a bivariate normal distribution. In the case of the mean error and the standard deviation of the mean error, the assumption is that the errors are normally distributed, which for continuous variables, is usually a reasonable assumption, even for the standard deviation of the errors.

Bootstrap confidence intervals for any verification statistic are available in MET. Bootstrapping is a nonparametric statistical method for estimating parameters and uncertainty information. The idea is to obtain a sample of the verification statistic(s) of interest (e.g., bias ETS, etc.) so that inferences can be made from this sample. The assumption is that the original sample of matched forecast-observation pairs is representative of the population. Several replicated samples are taken with replacement from this set of forecast-observation pairs of variables (e.g., precipitation, temperature, etc.), and the statistic(s) are calculated for for each replicate. That is, given a set of *n* forecast-observation pairs, we draw values at random from these pairs, allowing the same pair to be drawn more than once, and the statistic(s) is (are) calculated for each replicated sample. This yields a sample of the statistic(s) based solely on the data without making any assumptions about the underlying distribution of the sample. It should be noted, however, that if the observed sample of matched pairs is dependent, then this dependence should be taken into account somehow. Currently, in the confidence interval methods in MET do not take into account dependence, but future releases will support a robust method allowing for dependence in the original sample. More detailed information about the bootstrap algorithm is found in the appendix.

Confidence intervals can be calculated from the sample of verification statistics obtained through the bootstrap algorithm. The most intuitive method is to simply take the appropriate quantiles of the sample of statistic(s). For example, if one wants a 95% CI, then one would take the 2.5 and 97.5 percentiles of the resulting sample. This method is called the percentile method, and has some nice properties. However, if the original sample is biased and/or has non-constant variance, then it is well known that this interval is too optimistic. The most robust, accurate, and well-behaved way to obtain accurate CIs from bootstrapping is to use the bias corrected and adjusted percentile method (or BCa). If there is no bias, and the variance is constant, then this method will yield the usual percentile interval. The only drawback to the approach is that it is computationally intensive. Therefore, both the percentile and BCa methods are available in MET, with the considerably more efficient percentile method being the default.

The only other option associated with bootstrapping currently available in MET is to obtain replicated samples smaller than the original sample (i.e., to sample m<n points at each replicate). Ordinarily, one should use m=n, and this is the default. However, there are cases where it is more appropriate to use a smaller value of m (e.g., when making inference about high percentiles of the original sample). See Gilleland (2008) for more information and references about this topic.

MET version 1.1 provides parametric confidence intervals based on assumptions of normality for the following categorical statistics:

- Base Rate
- Forecast Mean
- Accuracy
- Probability of Detection
- Probability of Detection of the non-event
- Probability of False Detection
- False Alarm Ratio
- Critical Success Index
- Hanssen-Kuipers Discriminant
- Odds Ratio

MET version 1.1 provides parametric confidence intervals based on assumptions of normality for the following continuous statistics:

- Forecast and Observation Means
- Forecast, Observation, and Error Standard Deviations
- Pearson Correlation Coefficient
- Mean Error

MET version 1.1 provides non-parametric bootstrap confidence intervals for 13 categorical and 17 continuous statistics. Kendall's Tau and Spearman's Rank correlation coefficients are the only exceptions. Computing bootstrap confidence intervals for these statitistics would be computationally unrealistic.

For more information on confidence intervals pertaining to verification measures, 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 or in the NetCDF format that is output by the Pcp-Combine tool. The point observations must be in NetCDF format as the output of the `pb2nc` or `ascii2nc` 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
        config_file
        [-climo climo_file]
        [-ncfile netcdf_file]
        [-outdir path]
        [-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 indicates 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 verifying the model.

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

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

1. 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.

2. The **`-ncfile` netcdf_file** may be used to pass additional NetCDF point observation files to be used in the verification.

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

4. The **`-v` level** option indicates the desired level of verbosity. The value 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
               PointStatConfig
```

In this example, the Point-Stat tool evaluates the model data in the **`sample_fcst.grb`** GRIB file using the observations in the NetCDF output of `pb2nc`, **`sample_pb.nc`** applying the configuration options specified in the PointStatConfig file.

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. Each configuration file (both the default and sample) contains many comments describing its contents. The contents of the configuration file are also described in the subsections below.

Note that environment variables may be used when editing configuration files, as described in the section 3.5.2 for the PB2NC.

```
model = "WRF";
```

The **`model`** variable contains a short text string identifying the name to be assigned to the model being verified. This text string is written out as a header column of the VSDB output so that verification statistics from multiple models may be differentiated. The value listed above is simply set to "WRF".

```
vx_grib_code[]    =    [“SPFH/P500”,    “TMP/P500”,    “HGT/P500”,
“UGRD/P500”, “VGRD/P500”];
```

The **`vx_grib_code`** variable contains a comma-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. A level indicator in the form “**ANNN**”, “**ZNNN**”, “**PNNN**”, or “**PNNN-NNN**” must follow each GRIB code. These indicate an accumulation interval, a single vertical level, a single pressure level, and a range of pressure levels, respectively. “**NNN**” indicates the accumulation or level value. The values listed above indicate that specific humidity, temperature, height, and the U and V components of the winds should all be verified 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 to indicate greater than, greater than or equal to, equal to, not equal to, less than, and less than or equal to, respectively. The number of entries in **`thresholds`** must match the number of entries in **`vx_grib_code`**. However, multiple thresholds may be applied to each GRIB code by provided a space-separated list within the double quotes (e.g. “gt0 le 0”). It is the user's responsibility to know the units for each model variable and to choose appropriate threshold values.

```
message_type[] = ["ADPUPA"];
```

The Point-Stat tool performs verification using observations for one message type at a time. The **`message_type`** variable contains a comma-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 comma-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. Supplying a value of “**FULL**“

indicates that the verification should be performed over the entire grid on which the data resides. The value listed above indicates that verification should be performed 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 comma-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.

```
mask_stations = "";
```

The **`mask_stations`** variable contains a filename which contains a space-separated list of station ID's at which verification should be performed.

```
ci_alpha[] = [0.05];
```

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

```
boot_interval = 1;
```

The **`boot_interval`** variable indicates what method should be used for computing bootstrap confidence intervals. A value of 0 indicates that the highly accurate but computationally intensive BCa (bias-corrected percentile) method should be used. A value of 1 indicates that the somewhat less accurate but efficient percentile method should be used.

```
boot_rep_prop = 1.0;
```

The **`boot_rep_prop`** variable must be set to a value between 0 and 1. When computing bootstrap confidence intervals over n sets of matched pairs, the size of the subsample, m, may be chosen less than or equal to the size of the sample, n. This variable defines the size of m as a proportion relative to the size of n. A value of 1, as shown above, indicates that the size of the subsample, m, should be equal to the size of the sample, n.

```
n_boot_rep = 1000;
```

The **n_boot_rep** variable defines the number of subsamples that should be taken when computing bootstrap confidence intervals. This variable should be set large enough so that when confidence intervals are computed multiple times for the same set of data, the intervals do not change much. Setting this variable to zero disables the computation of bootstrap confidence intervals which may be necessary to run in realtime or near-realtime over large domains. Setting this variable to 1000, as shown above, indicates that bootstrap confidence interval should be computed over 1000 subsamples of the matched pairs.

```
boot_rng = "mt19937";
```

The **boot_rng** variable defines the random number generator to be used in the computation of bootstrap confidence intervals. Subsamples are chosen at random from the full set of matched pairs. The randomness is determined by the random number generator specified. Users should refer to detailed documentation of the GNU Scientific Library for a listing of the random number generators available for use.

```
boot_seed = "";
```

The **boot_seed** variable may be set to a specific value to make the computation of bootstrap confidence intervals fully repeatable. When left empty, as show above, the random number generator seed is chosen automatically which will lead to slightly different bootstrap confidence intervals being computed each time the data is run. Specifying a value here ensures that the bootstrap confidence intervals will be computed the same over multiple runs of the same data.

```
interp_method[] = [ "DW_MEAN" ];
```

The **interp_method** variable contains a comma-separated list of interpolation methods to be used when interpolating forecast data to observation locations. The valid values which may be listed are MIN, MAX, MEDIAN, UW_MEAN, DW_MEAN, and LS_FIT for the minimum, maximum, median, unweighted mean, distance-weighted mean, and a least squares fit. Providing multiple interpolation methods indicates that statistics should be computed multiple times using a different interpolation method each time. These methods are described in Section 4.2.1.

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

The **`interp_width`** variable contains a comma-separated list of values to be used in defining the neighborhoods over which the interpolation is performed. The neighborhood is simply a 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 defines a 2 x 2 square of grid points around the observation point (the 4 closest model grid points), while an **`interp_with`** of 3 defines a 3 x 3 square of grid points around the observation point, and so on. The values listed above indicate that the nearest neighbor and the 4 closest grid points should be used to define the neighborhoods.

```
interp_threshold = 1.0;
```

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, 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 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
- MPR for Matched Pair data

Note that the first five line types are easily derived from one another. Users are 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.

```
rank_corr_flag = 1;
```

The **`rank_corr_flag`** variable may be set to 0 (“no”) or 1 (“yes”) to indicate whether or not Kendall’s Tau and Spearman’s Rank correlation coefficients should be computed. The computation of these rank correlation coefficients is very slow when run over many matched pairs. By default, this flag is turned on, as shown above, but setting it to 0 should improve the runtime performance.

```
version = “V1.1”;
```

The **`version`** indicates the version of the `point_stat` configuration file used. Future versions of MET may include changes to `point_stat` and the `point_stat` configuration file. This value should not be modified.

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_`HHMMSS`L`_YYYYMMDD_HHMMSS`V.vsdb`** where **YYYYMMDDHH indicates the forecast lead time and YYYYMMDD_HHMMSS** indicates the forecast valid time. The output ASCII files are named similarly: **`point_stat_`HHMMSS`L`_YYYYMMDD_HHMMSS`V`_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 10 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.

.

Table 4-2. *Header information for each file point-stat outputs.*

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	FCST_LEAD	Forecast lead time in HHMMSS format
4	FCST_VALID	Forecast valid 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, Threshold, and/or Alpha Value*	Varies by line type, see tables below
8	VAR	Model variable being verified
9	LEVEL	Vertical level at which verification performed
10	=	Equal sign delimeter between the header columns and the data columns

Table 4-3. *Format information for FHO (Forecast, Hit rate, Observation rate) output line type.*

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

Table 4-4. *Format information for CTC (Contingency Table Count) output line type.*

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

Table 4-5. *Format information for CTP (Contingency Table Proportion) output line type.*

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

Table 4-6. Format information for CFP (Contingency Forecast Proportion) output line type.

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

Table 4-7. Format information for COP (Contingency Observation Proportion) output line type.

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

Table 4-8. *Format information for CTS (Contingency Table Statistics) output line type.*

CTS OUTPUT FORMAT		
Column Number	**CTS Column Name**	**Description**
7	`CTS_T/ALPHA`	Contingency Table Statistics line type, threshold applied, and alpha value used in confidence intervals
11	`TOTAL`	Total number of matched pairs
12-16	`BASER, BASER_NCL, BASER_NCU, BASER_BCL, BASER_BCU`	Base rate including normal and bootstrap upper and lower confidence limits
17-21	`FMEAN, FMEAN_NCL, FMEAN_NCU, FMEAN_BCL, FMEAN_BCU,`	Forecast mean including normal and bootstrap upper and lower confidence limits
22-26	`ACC, ACC_NCL, ACC_NCU, ACC_BCL, ACC_BCU`	Accuracy including normal and bootstrap upper and lower confidence limits
27-29	`FBIAS, FBIAS_BCL, FBIAS_BCU`	Frequency Bias including bootstrap upper and lower confidence limits
30-34	`PODY, PODY_NCL, PODY_NCU, PODY_BCL, PODY_BCU`	Probability of detecting yes including normal and bootstrap upper and lower confidence limits
35-39	`PODN, PODN_NCL, PODN_NCU, PODN_BCL, PODN_BCU`	Probability of detecting no including normal and bootstrap upper and lower confidence limits
40-44	`POFD, POFD_NCL, POFD_NCU, POFD_BCL, POFD_BCU`	Probability of false detection including normal and bootstrap upper and lower confidence limits
45-49	`FAR, FAR_NCL, FAR_NCU, FAR_BCL, FAR_BCU`	False alarm ratio including normal and bootstrap upper and lower confidence limits
50-54	`CSI, CSI_NCL, CSI_NCU, CSI_BCL, CSI_BCU`	Critical Success Index including normal and bootstrap upper and lower confidence limits

CTS OUTPUT FORMAT		
Column Number	**CTS Column Name**	**Description**
55-57	GSS, GSS_BCL, GSS_BCU	Gilbert Skill Score including bootstrap upper and lower confidence limits
58-62	HK, HK_NCL, HK_NCU, HK_BCL, HK_BCU	Hanssen-Kuipers Discriminant including normal and bootstrap upper and lower confidence limits
63-65	HSS, HSS_BCL, HSS_BCU	Heidke Skill Score.including bootstrap upper and lower confidence limits
66-70	ODDS, ODDS_NCL, ODDS_NCU, ODDS_BCL, ODDS_BCU	Odds Ratio including normal and bootstrap upper and lower confidence limits
71	INTERP_MTHD	Interpolation method applied
72	INTERP_PNTS	Number of points in the interpolation neighborhood

***Table 4-9.** Format information for CNT(Continuous Statistics) output line type.*

CNT OUTPUT FORMAT		
Column Number	**CNT Column Name**	**Description**
7	CNT/ALPHA	Continuous statistics line type and alpha value applied
11	TOTAL	Total number of matched pairs
12-16	FBAR, FBAR_NCL, FBAR_NCU, FBAR_BCL, FBAR_BCU	Forecast mean including normal and bootstrap upper and lower confidence limits
17-21	FSTDEV, FSTDEV_NCL, FSTDEV_NCU, FSTDEV_BCL, FSTDEV_BCU	Standard deviation of the forecasts including normal and bootstrap upper and lower confidence limits
22-26	OBAR, OBAR_NCL, OBAR_NCU, OBAR_BCL, OBAR_BCU	Observation mean including normal and bootstrap upper and lower confidence limits
27-31	OSTDEV, OSTDEV_NCL, OSTDEV_NCU, OSTDEV_BCL, OSTDEV_BCU	Standard deviation of the observations including normal and bootstrap upper and lower confidence limits

CNT OUTPUT FORMAT		
Column Number	**CNT Column Name**	**Description**
32-36	`PR_CORR`, `PR_CORR_NCL`, `PR_CORR_NCU`, `PR_CORR_BCL`, `PR_CORR_BCU`	Pearson correlation coefficient including normal and bootstrap upper and lower confidence limits
37	`SP_CORR`	Spearman's rank correlation coefficient
38	`KT_CORR`	Kendall's tau statistic
39	`RANKS`	Number of ranks used in computing Kendall's tau statistic
40	`FRANK_TIES`	Number of tied forecast ranks used in computing Kendall's tau statistic
41	`ORANK_TIES`	Number of tied observation ranks used in computing Kendall's tau statistic
42-46	`ME`, `ME_NCL`, `ME_NCU`, `ME_BCL`, `ME_BCU`	Mean error (F-O) including normal and bootstrap upper and lower confidence limits
47-51	`ESTDEV`, `ESTDEV_NCL`, `ESTDEV_NCU`, `ESTDEV_BCL`, `ESTDEV_BCU`	Standard deviation of the error including normal and bootstrap upper and lower confidence limits
52-54	`MBIAS`, `MBIAS_BCL`, `MBIAS_BCU`	Multiplicative bias including bootstrap upper and lower confidence limits
55-57	`MAE`, `MAE_BCL`, `MAE_BCU`	Mean absolute error including bootstrap upper and lower confidence limits
58-60	`MSE`, `MSE_BCL`, `MSE_BCU`	Mean squared error including bootstrap upper and lower confidence limits
61-63	`BCMSE`, `BCMSE_BCL`, `BCMSE_BCU`	Bias-corrected mean squared error including bootstrap upper and lower confidence limits
64-66	`RMSE`, `RMSE_BCL`, `RMSE_BCU`	Root mean squared error including bootstrap upper and lower confidence limits

CNT OUTPUT FORMAT		
Column Number	CNT Column Name	Description
67-81	E10, E10_BCL, E10_BCU, E25, E25_BCL, E25_BCU, E50, E50_BCL, E50_BCU, E75, E75_BCL, E75_BCU, E90, E90_BCL, E90_BCU	10th, 25th, 50th, 75th, and 90th percentiles of the error including bootstrap upper and lower confidence limits
82	INTERP_MTHD	Interpolation method applied
83	INTERP_PNTS	Number of points in the interpolation neighborhood

Table 4-10. *Format information for SL1L2 (Scalar Partial Sums) output line type.*

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

Table 4-11. *Format information for SAL1L2 (Scalar Anomaly Partial Sums) output line type.*

SAL1L2 OUTPUT FORMAT		
Column Number	SAL1L2 Column Name	Description
7	SAL1L2	Scalar Anomaly L1L2 line type
11	TOTAL	Total number of matched triplets of forecast (f), observation (o), and climatological value (c)
12	FABAR	Mean(f-c)
13	OABAR	Mean(o-c)

SAL1L2 OUTPUT FORMAT		
Column Number	**SAL1L2 Column Name**	**Description**
14	`FOABAR`	Mean((f-c)*(o-c))
15	`FFABAR`	Mean($(f\text{-}c)^2$)
16	`OOABAR`	Mean($(o\text{-}c)^2$)
17	`INTERP_MTHD`	Interpolation method applied
18	`INTERP_PNTS`	Number of points in the interpolation neighborhood

Table 4-12. *Format information for VL1L2 (Vector Partial Sums) output line type.*

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

Table 4-13. *Format information for VAL1L2 (Vector Anomaly Partial Sums) output line type.*

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

Table 4-14. *Format information for MPR (Matched Pair) output line type.*

MPR OUTPUT FORMAT		
Column Number	**MPR Column Name**	**Description**
7	`MPR`	Matched Pair line type
11	`TOTAL`	Total number of matched pairs
12	`INDEX`	Index for the current matched pair
13	`OBS_LAT`	Latitude of the observation in degrees north
14	`OBS_LON`	Longitude of the observation in degrees east
15	`OBS_LVL`	Pressure level of the observation in hPa or accumulation interval in hours
16	`OBS_ELV`	Elevation of the observation in meters above sea level
17	`FCST`	Forecast value interpolated to the observation location
18	`OBS`	Observation value
19	`CLIMO`	Climatological value
20	`INTERP_MTHD`	Interpolation method applied
21	`INTERP_PNTS`	Number of points in the interpolation neighborhood

The VSDB output files described for `point_stat` may be used as inputs to the VSDB Analysis tool. For more information on using the VSDB Analysis tool to create stratifications and aggregations of the VSDB files produced by `point_stat`, please see Chapter 7.

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 compute the verification statistics. The Grid-Stat tool functions in much the same way as the Point-Stat tool, except that no interpolation is required because the forecasts and observations are on the same grid. However, the interpolation parameters may be used to perform a smoothing operation on the forecast field prior to verifying it. In addition to traditional verification approaches, the Grid-Stat tool includes neighborhood methods, designed to examine forecast performance as a function of spatial scale.

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

5.2 Scientific and statistical aspects

5.2.1 Statistical measures

The Grid-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. These categories of measures are briefly described here; specific descriptions of the measures are provided in Appendix C. Additional information 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 partial sums and other FHO statistics that are produced by the NCEP verification system. These statistics are also described in Appendix C.

Measures for categorical variables
Categorical verification statistics are used to evaluate forecasts that are in the form of a discrete set of categories rather than on a continuous scale. Currently, Point-Stat computes categorical 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 < 273.15K). See Appendix C for more information.

Measures for continuous variables
For continuous variables, many verification measures are based on the forecast error (i.e., $f - o$). However, it also is of interest to investigate characteristics of the forecasts, and the observations, as well as their relationship. These concepts are consistent with

the general framework for verification outlined by Murphy and Winkler (1987). The statistics produced by MET for continuous forecasts represent this philosophy of verification, which focuses on a variety of aspects of performance rather than a single measure. See Appendix C for specific information.

Use of analysis fields for verification

The Grid-Stat tool allows evaluation of model forecasts using model analysis fields. However, users are cautioned that an analysis field is not independent of its parent model; for this reason verification of model output using an analysis field from the same model is generally not recommended and is not likely to yield meaningful information about model performance.

5.2.2 Statisitcal confidence intervals

The confidence intervals for the Grid-Stat tool are the same as those provided for the Point-Stat tool except that the scores are based on pairing grid points with grid points so that there are likely more values for each field making any assumptions based on the central limit theorem more likely to be valid. However, it should be noted that spatial (and temporal) correlations are not presently taken into account in the confidence interval calculations. Therefore, confidence intervals reported may be somewhat too narrow (e.g., Efron 2007). See Appendix D for details regarding confidence intervals provided by MET.

5.2.3 Neighborhood methods

MET also incorporates several neighborhood methods to give credit to forecasts that are close to the observations, but not necessarily exactly matched up in space. Also referred to as “fuzzy” verification methods, these methods do not just compare a single forecast at each grid point to a single observation at each grid point; they compare the forecasts and observations in a neighborhood surrounding the point of interest. With the neighborhood method, the user chooses a distance within which the forecast event can fall from the observed event and still be considered a hit. In MET this is implemented by defining a square search window around each grid point. Within the search window, the number of observed events is compared to the number of forecast events. In this way, credit is given to forecasts that are close to the observations without requiring a strict match between forecasted events and observed events at any particular grid point. The neighborhood methods allow the user to see how forecast skill varies with neighborhood size and can help determine the smallest neighborhood size that can be used to give sufficiently accurate forecasts.

There are several ways to present the results of the neighborhood approaches, such as the Fractions Skill Score (FSS) or the Fractions Brier Score (FBS). These scores are presented in Appendix C. One can also simply up-scale the information on the forecast verification grid by smoothing or resampling within a specified neighborhood around each grid point and recalculate the traditional verification metrics on the coarser grid. The MET output includes traditional contingency table statistics for each threshold and neighborhood window size.

The user must specify several parameters in the **`grid_stat`** configuration file to utilize the neighborhood approach, such as the interpolation method, size of the smoothing window, and required fraction of valid data points within the smoothing window. For FSS-specific results, the user must specify the size of the neighborhood window, the required fraction of valid data points within the window, and the fractional coverage threshold from which the contingency tables are defined. These parameters are described further in the practical information section below.

5.3 Practical information

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

The Grid-Stat tool provides the capability of verifying one or more model variables/levels using multiple thresholds for each model variable/level. The Grid-Stat tool performs no interpolation because the input model and observation datasets must already be on a common grid. However, the interpolation parameters may be used to perform a smoothing operation on the forecast field prior to verifying it to investigate how the scale of the forecast affects the verification statistics. The Grid-Stat tool computes a number of continuous statistics for the forecast minus observation differences 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
        config_file
        [-outdir path]
        [-v level]
```

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

Required arguments for **`grid_stat`**

1. The **fcst_file** argument indicates 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 **config_file** argument indicates the name of the configuration file to be used. The contents of the configuration file are discussed below.

Optional arguments for **`grid_stat`**

1. The **`-outdir`** **path** indicates the directory where output files should be written.

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.

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

Example 1:
```
grid_stat sample_fcst.grb
          sample_obs.grb
          GridStatConfig
```

In Example 1, the Grid-Stat tool will verify the model data in the **`sample_fcst.grb`** GRIB file using the observations in the **`sample_obs.grb`** GRIB file applying the configuration options specified in the **`GridStatConfig`** file.

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

Example 2:
```
grid_stat sample_fcst.nc
          sample_obs.nc
          GridStatConfig
```

In the second example, the Grid-Stat tool will verify the model data in the **`sample_fcst.nc`** NetCDF output of `pcp_combine`, using the observations in the **`sample_obs.nc`** NetCDF output of `pcp_combine`, and applying the configuration options specified in the **`GridStatConfig`** file. Because the model and observation files contain only a single field of accumulated precipitation, the **`GridStatConfig`** file should be configured to specify that only accumulated precipitation be verified.

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 default configuration file contains many comments describing its contents. The contents are also described in more detail below.

Note that environment variables may be used when editing configuration files, as described in the section 3.5.2 for the PB2NC.

```
model = "WRF";
```

The **`model`** variable contains a short text string identifying the name to be assigned to the model being verified. This text string is written out as a header column of the VSDB output so that verification statistics from multiple models may be differentiated. The value listed above is simply set to "WRF".

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

The **`vx_grib_code`** variable contains a comma-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 "**`A`NNN**", "**`Z`NNN**", "**`P`NNN**", or "**`P`NNN-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. The value listed above indicates that accumulated precipitation (GRIB code 61) with a 3-hourly accumulation interval should be verified. 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`** to indicate greater than, greater than or equal to, equal to, not equal to, less than, and less than or equal to, respectively. The number of entries in **`thresholds`** must match the number of entries in **`vx_grib_code`**. However, multiple thresholds may be applied to each GRIB code by provided a space-separated list within the double quotes. The values listed above indicate that 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 comma-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. Supplying a value of “**FULL**“ indicates that the verification should be performed over the entire grid on which the data resides. The value listed above indicates that verification should be performed 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 comma-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.

```
ci_alpha[] = [0.05];
```

The **`ci_alpha`** variable contains a comma-separated list of alpha values to be used when computing confidence intervals. The confidence interval computed is 1 minus the **`ci_alpha`** value. The value of 0.05 listed above indicates 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_method[] = [ ”DW_MEAN” ];
```

The **`interp_method`** variable contains a comma-separated list of operations to be performed on the forecast field prior to performing verification. The valid values which may be listed are **MIN**, **MAX**, **MEDIAN**, and **UW_MEAN** for the minimum, maximum, median, and unweighted mean. Providing multiple interpolation methods indicates that statistics should be computed multiple times performing different smoothing operations each time. These methods are described in Section 4.2.1.

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

The **`interp_width`** variable contains a comma-separated list of values to be used in defining the neighborhoods over which the smoothing operation is performed on the forecast field. The neighborhood is simply a square centered on the observation point. The **`interp_width`** value specifies the width of that square.

```
interp_threshold = 1.0;
```

The **`interp_threshold`** variable contains a number between 0 and 1. When performing a smoothing operation 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, no smoothed value is compuated. 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.

```
nbr_width[] = [3, 5];
```

The **`nbr_width`** variable contains a comma-separated list of values to be used in defining the neighborhood size to be used when computing neighborhood verification statistics. The neighborhood is simply a square centered on the current point and the **`nbr_width`** value specifies the width of that square.

```
nbr_threshold = 1.0;
```

The **`nbr_threshold`** variable contains a number between 0 and 1. When performing neighborhood verification over some neighborhood of points the ratio of the number of valid data points to the total number of points in the neighborhood is computed. If that ratio is greater than this threshold, that value is not included in the neighborhood verification. Setting this threshold to 1, which is the default, requires that the entire neighborhood must contain valid data. This variable will typically come into play only along the boundaries of the verification region chosen.

```
nbr_frac_threshold[] = ["ge0.5"];
```

The **`nbr_frac_threshold`** variable contains a comma sparated list of thresholds to be applied to the neighborhood coverage field. When performing neighborhood verification, forecast and observation coverage fields are computed from the forecast and observation raw fields using the raw threshold value (**`thresholds`** variable) and the neighborhood size (**`nbr_width`** variable). Those coverage fields may then be thresholded using the values specified for this variable to create 0/1 mask fields. The mask fields are compared point by point to compute a contingency table and the corresponding contingency table statistcs.

```
output_flag[] = [ 2, 2, 2, 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
9. NBRCTC for Neighborhood Contingency Table Counts
10. NBRCTS for Neighborhood Contingency Table Statistics
11. NBRCNT for Neighborhood Continuous Statistics

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 matched pair and forecast minus observation difference fields 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 disables its creation.

```
rank_corr_flag = 1;
```

The **`rank_corr_flag`** variable may be set to 0 (“no”) or 1 (“yes”) to indicate whether or not Kendall’s Tau and Spearman’s Rank correlation coefficients should be computed. The computation of these rank correlation coefficients is very slow when run over many matched pairs. By default, this flag is turned on, as shown above, but setting it to 0 should improve the runtime performance.

```
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.

```
version = “V1.1”;
```

The **`version`** indicates the version of the `grid_stat` configuration file used. Future versions of MET may include changes to `grid_stat` and the `grid_stat` configuration file. This value should not be modified.

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 are written to the default output directory or the directory specified by the **`-outdir`** command-line option.

The output VSDB file is named using the following naming convention: **`grid_stat_`HHMMSS`L_`YYYYMMDD_HHMMSS`V.vsdb`** where **HHMMSS** indicates the forecast lead time and and **YYYYMMDD_HHMMSS** indicates the forecast valid time.

The output ASCII files are named similarly: **`grid_stat_`HHMMSS`L_`YYYYMMDD_HHMMSS`V_`TYPE`.txt`** where **TYPE** is one of **`fho`**, **`ctc`**, **`ctp`**, **`cfp`**, **`cop`**, **`cts`**, **`cnt`**, or **`sl1l2`** to indicate the line type it contains.

The format of the VSDB and ASCII output of the Grid-Stat tool the same as the format of the VSDB and ASCII output of the Point-Stat tool with the exception of the three additional neighborhood line types. Please refer to the tables in section 4.3.3 (`point_stat` output) for a description of the common output VSDB and optional ASCII file line types. The formats of the three additional neighborhood line types for `grid_stat` are explained in the following tables.

Table 5-1. *Header information for each file grid-stat outputs.*

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	`FCST_LEAD`	Forecast lead time in HHMMSS format
4	`FCST_VALID`	Forecast valid time in YYYYMMDDHH format
5	`OBTYPE`	Verifying observation type

HEADER		
Column Number	**Header Column Name**	**Description**
6	`VX_MASK`	Verifying masking region indicating the masking grid or polyline region applied
7	*`Line Type, Threshold, and/or Alpha Value`*	Varies by line type, see tables below
8	`VAR`	Model variable being verified
9	`LEVEL`	Vertical level at which verification performed
10	`=`	Equal sign delimeter between the header columns and the data columns

Table 5-2. *Format information for NBRCTC (Neighborhood Contingency Table Counts) output line type.*

NBRCTC OUTPUT FORMAT		
Column Number	**NBRCTC Column Name**	**Description**
7	`NBRCTC_T1_T2`	Neighborhood Contingency Table Counts line type, raw threshold applied, and coverage field threshold applied
11	`TOTAL`	Total number of matched pairs
12	`FY_OY`	Number of forecast yes and observation yes
13	`FY_ON`	Number of forecast yes and observation no
14	`FN_OY`	Number of forecast no and observation yes
15	`FN_ON`	Number of forecast no and observation no
16	`INTERP_PNTS`	Number of points in the interpolation neighborhood

Table 5-3. *Format information for NBRCTS (Neighborhood Contingency Table Statistics) output line type.*

NBRCTS OUTPUT FORMAT		
Column Number	**NBRCTS Column Name**	**Description**
7	`CTS_T1_T2/ALPHA`	Neighborhood Contingency Table Statistics line type, raw threshold applied, coverage field threshold applied, and alpha value used in confidence intervals
11	`TOTAL`	Total number of matched pairs
12-16	`BASER, BASER_NCL, BASER_NCU, BASER_BCL, BASER_BCU`	Base rate including normal and bootstrap upper and lower confidence limits

NBRCTS OUTPUT FORMAT		
Column Number	**NBRCTS Column Name**	**Description**
17-21	`FMEAN, FMEAN_NCL, FMEAN_NCU, FMEAN_BCL, FMEAN_BCU,`	Forecast mean including normal and bootstrap upper and lower confidence limits
22-26	`ACC, ACC_NCL, ACC_NCU, ACC_BCL, ACC_BCU`	Accuracy including normal and bootstrap upper and lower confidence limits
27-29	`FBIAS, FBIAS_BCL, FBIAS_BCU`	Frequency Bias including bootstrap upper and lower confidence limits
30-34	`PODY, PODY_NCL, PODY_NCU, PODY_BCL, PODY_BCU`	Probability of detecting yes including normal and bootstrap upper and lower confidence limits
35-39	`PODN, PODN_NCL, PODN_NCU, PODN_BCL, PODN_BCU`	Probability of detecting no including normal and bootstrap upper and lower confidence limits
40-44	`POFD, POFD_NCL, POFD_NCU, POFD_BCL, POFD_BCU`	Probability of false detection including normal and bootstrap upper and lower confidence limits
45-49	`FAR, FAR_NCL, FAR_NCU, FAR_BCL, FAR_BCU`	False alarm ratio including normal and bootstrap upper and lower confidence limits
50-54	`CSI, CSI_NCL, CSI_NCU, CSI_BCL, CSI_BCU`	Critical Success Index including normal and bootstrap upper and lower confidence limits
55-57	`GSS, GSS_BCL, GSS_BCU`	Gilbert Skill Score including bootstrap upper and lower confidence limits
58-62	`HK, HK_NCL, HK_NCU, HK_BCL, HK_BCU`	Hanssen-Kuipers Discriminant including normal and bootstrap upper and lower confidence limits
63-65	`HSS, HSS_BCL, HSS_BCU`	Heidke Skill Score.including bootstrap upper and lower confidence limits

NBRCTS OUTPUT FORMAT		
Column Number	**NBRCTS Column Name**	**Description**
66-70	`ODDS`, `ODDS_NCL`, `ODDS_NCU`, `ODDS_BCL`, `ODDS_BCU`	Odds Ratio including normal and bootstrap upper and lower confidence limits
71	`INTERP_PNTS`	Number of points in the interpolation neighborhood

Table 5-4. *Format information for NBRCNT(Neighborhood Continuous Statistics) output line type.*

NBRCNT OUTPUT FORMAT		
Column Number	**NBRCNT Column Name**	**Description**
7	`CNT_T/ALPHA`	Neighborhood Continuous statistics line type, raw threshold applied, and alpha value used in confidence intervals
11	`TOTAL`	Total number of matched pairs
12-14	`FBS`, `FBS_BCL`, `FBS_BCU`	Fractions Brier Score including bootstrap upper and lower confidence limits
15-17	`FSS`, `FSS_BCL`, `FSS_BCU`	Fractions Skill Score including bootstrap upper and lower confidence limits
18	`INTERP_PNTS`	Number of points in the interpolation neighborhood

If requested in the **`output_flag`** array, a NetCDF file containing the matched pair and forecast minus observation difference fields for each combination of variable type/level and masking region applied will be generated. The output NetCDF file is named similarly to the other output files: **`grid_stat_`HHMMSS`L`_YYYYMMDD_HHMMSS`V_pairs.nc`**. Commonly available NetCDF utilities such as `ncdump` or `ncview` may be used to view the contents of the output file.

The output NetCDF file contains the dimensions and variables shown in the following Tables 5-5 and 5-6.

Table 5-5. *Dimensions defined in NetCDF matched pair output.*

`grid_stat` ***NetCDF DIMENSIONS***	
NetCDF Dimension	**Description**
`Lat`	Dimension of the latitude (i.e. Number of grid points in the North-South direction)
`Lon`	Dimension of the longitude (i.e. Number of grid points in the East-West direction)

Table 5-6. *Variables defined in NetCDF matched pair output.*

`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
`FCST_VAR_LVL_MASK _INTERP_MTHD _INTERP_PNTS`	lat, lon	For each model variable (VAR), vertical level (LVL), masking region (MASK), and, if applicable, smoothing operation (INTERP_MTHD and INTERP_PNTS), the forecast value is listed for each point in the mask
`DIFF_VAR_LVL_MASK _INTERP_MTHD _INTERP_PNTS`	lat, lon	For each model variable (VAR), vertical level (LVL), masking region (MASK), and, if applicable, smoothing operation (INTERP_MTHD and INTERP_PNTS), the difference (forecast – observation) is computed for each point in the mask
`OBS_VAR_LVL_MASK`	lat, lon	For each model variable (VAR), vertical level (LVL), and masking region (MASK), the observation value is listed for each point in the mask

The VSDB output files described for `grid_stat` may be used as inputs to the VSDB Analysis tool. For more information on using the VSDB Analysis tool to create stratifications and aggregations of the VSDB files produced by `grid_stat`, please see Chapter 7.

Chapter 6 – The MODE Tool

6.1 Introduction

This chapter provides a description 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 at http://www.rap.ucar.edu/projects/icp/index.html.

MODE may be used in a generalized way to compare any two fields. For simplicity, field1 may be thought of in this Chapter as “the forecast,” while field2 may be thought of as “the observation”, which is usually a gridded analysis of some sort. The convention of field1/field1 is also used in Table 6-2. MODE resolves objects in both the forecast and observed fields. These objects mimic what humans would call “regions of interest”. Object attributes are calculated and compared, and are used to associate (“merge”) objects within a single field, as well as to “match” objects between the forecast and observed fields. Finally, summary statistics describing the objects and object pairs are produced. These statistics can be used to identify correlations and differences among the objects, leading to insights concerning forecast strengths and weaknesses.

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

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

The parameters R and H are not independent. They are related by the requirement that the integral of ϕ over the grid be unity:

$$\pi R^2 H = 1.$$

Thus, the radius of influence R is the only tunable parameter in the convolution process. Once R is chosen, H is determined by the above equation.

Once the convolved field, C, is in hand, it is thresholded to create a mask field, M:

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

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

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

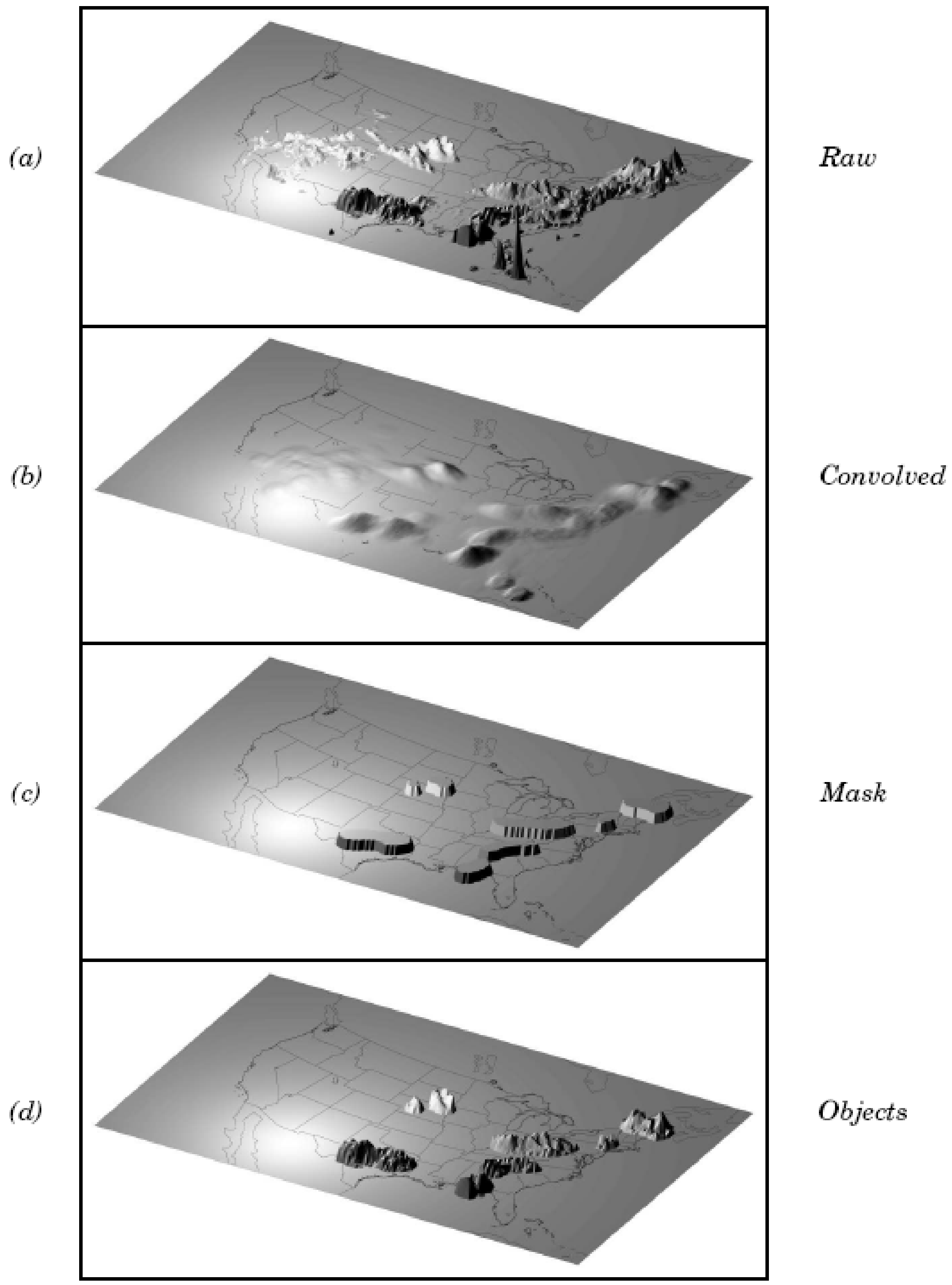


Figure 6-1: *Example of an application of the MODE object identification process to a model precipitation field.*

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 a large, extended object.

Axis Angle, denoted by θ, is calculated from the second-order moments. It gives information on the orientation or “tilt” of an object. *Curvature* is another attribute that uses moments in its calculation, specifically, third-order moments.

Aspect Ratio is computed by fitting a rectangle around an object. The rectangle is aligned so that it has the same axis angle as the object, and the length and width are chosen so as to just enclose the object. We make no claim that the rectangle so obtained is the smallest possible rectangle enclosing the given object. However, this rectangle is much easier to calculate than a smallest enclosing rectangle and serves our purposes just as well. Once the rectangle is determined, the aspect ratio of the object is defined to be the width of the rectangle divided by its length.

Another object attribute defined by MODE is *complexity*. Complexity is defined by comparing the area of an object to the area of its convex hull.

All the attributes discussed so far are defined for *single* objects. Once these are determined, they can be used to calculate attributes for pairs of objects. One example is *centroid difference*. This measure is simply the (vector) difference between the centroids of the two objects. Another example is *angle difference*, the difference between the axis angles.

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

6.2.3 Fuzzy logic

Once object attributes $\alpha_1, \alpha_2, ..., \alpha_n$ are estimated, some of them are used as input to a fuzzy logic engine that performs the matching and merging steps. *Merging* refers to grouping together objects in a single field, while *matching* refers to grouping together objects in different fields, typically the forecast and observed fields. *Interest maps*, I_i, are applied to the individual attributes, α_i, to convert them into interest values, which range from zero (representing no interest) to one (high interest). For example, the default interest map for centroid difference is one for small distances, and falls to zero as the distance increases. For other attributes (e.g., intersection area), low values indicate low interest, and high values indicate more interest.

The next step is to define *confidence maps*, C_i, for each attribute. These maps (again with values ranging from zero to one) reflect how confident we are in the calculated value of an attribute. The confidence maps generally are functions of the entire attribute vector $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$, in contrast to the interest maps, where each I_i is a function only of α_i. To see why this is necessary, imagine an electronic anemometer that outputs a stream of numerical values of wind speed and direction. It is typically the case for such devices that when the wind speed becomes small enough, the wind direction is poorly resolved. The wind must be at least strong enough to overcome friction and turn the anemometer. Thus, in this case, our confidence in one attribute (wind direction) is dependent on the value of another attribute (wind speed). In MODE, all of the confidence maps except the map for axis angle are set to a constant value of 1. The axis angle confidence map is a function of aspect ratio, with values near one having low confidence, and values far from one having high confidence.

Next, scalar *weights*, w_i, are assigned to each attribute, representing an empirical judgment regarding the relative importance of the various attributes. As an example, in initial applications of MODE, centroid distance was weighted more heavily than other attributes, because the location of storm systems close to each other in space seemed to be a strong indication (stronger than that given by any other attribute) that they were related.

Finally, all these ingredients are collected into a single number called the *total interest*, T, given by

$$T(\alpha) = \frac{\sum_i w_i C_i(\alpha) I_i(\alpha_i)}{\sum_i w_i C_i(\alpha)}$$

This total interest value is then thresholded, and pairs of objects that have total interest values above the threshold are merged (if they are in the same field) or matched (if they are in different fields).

Another merging method is available in MODE, which can be used instead of, or along with, the fuzzy logic based merging just described. Recall that the convolved field was thresholded to produce the mask field. A second (lower) threshold can be specified so that objects that are separated at the higher threshold but joined at the lower threshold are merged.

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 9.) Refer to Section 6.3.3 for lists of the statistics included in the `mode` output files. Example scripts in R and IDL will be posted on the MET website in the future.

6.3 Practical information

This section contains a description of how MODE can be configured and run. The MODE tool is used to perform a features-based verification of gridded model data using gridded observations. The input gridded model and observation datasets must be in 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 analysis data may be based on observations, such as Stage II or Stage IV data for verifying accumulated precipitation, or a model analysis field may be used. However, users are cautioned that it is generally unwise to verify model output using an analysis field produced by the same model.

MODE provides the capability to select a single model variable/level from which to derive objects to be analyzed. MODE was developed and tested using accumulated precipitation. However, the code has been generalized to allow the use of any gridded model and observation field. Based on the options specified in the configuration file, MODE will define a set of simple objects in the model and observation fields. It will then compute an interest value for each pair of objects across the fields using a fuzzy engine approach. Those interest values are thresholded, and any pairs of objects above the threshold will be matched/merged. Through the configuration file, MODE offers a wide range of flexibility in how the objects are defined, processed, matched, and merged.

6.3.1 `mode` *usage*

The usage statement for the MODE tool is listed below:

```
Usage: mode
        fcst_file
        obs_file
        config_file
        [-config_merge merge_config_file]
        [-outdir path]
        [-plot]
        [-obj_plot]
        [-obj_stat]
        [-ct_stat]
        [-v level]
```

The MODE tool has three required arguments and can accept up to seven optional arguments.

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.

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

Optional arguments for **`mode`**

1. 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.

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

3. The **`-plot`** option disables the generation of the output PostScript plot containing a summary of the features-based verification technique.

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

5. 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.

6. 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.

7. 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 MODE calling sequence is listed below:

Example 1
```
mode      sample_fcst.grb
          sample_obs.grb
          WrfModeConfig_grb
```

In Example 1, the MODE tool will verify the model data in the **`sample_fcst.grb`** GRIB file using the observations in the **`sample_obs.grb`** GRIB file applying the configuration options specified in the **`WrfModeConfig_grb`** file.

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

Example 2
```
mode      sample_fcst.nc
          sample_obs.nc
          WrfModeConfig_nc
```

In Example 2, the MODE tool will verify the model data in the **`sample_fcst.nc`** NetCDF output of `pcp_combine` using the observations in the **`sample_obs.nc`** NetCDF output of `pcp_combine`, using the configuration options specified in the **`WrfModeConfig`** file. Since the model and observation files contain only a single field of accumulated precipitation, the **`WrfModeConfig_nc`** file should specify that accumulated precipitation be verified.

6.3.2 `mode` configuration file

The default configuration file for the MODE tool, **`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. . 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.

Note that environment variables may be used when editing configuration files, as described in the section 3.5.2 for the PB2NC.

```
model = “WRF”;
```

The **`model`** variable contains a short text string identifying the name to be assigned to the model being verified. This text string is written out in the first column of the ASCII output so that verification statistics from multiple models may be differentiated. The value listed above is simply set to “WRF”.

```
grid_res = 4;
```

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

```
fcst_grib_code = "APCP/A3" ;
obs_grib_code  = "APCP/A3" ;
```

The **`fcst_grib_code`** and **`obs_grid_code`** variables specify the model variable and observation variable, respectively, and the corresponding to the 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. A slash and a level indicator in the form “**`A`NNN**”, “**`Z`NNN**”, or “**`P`NNN**” must follow each GRIB code. The level indicators correspond to an accumulation interval (A), a single vertical level (Z), or a single pressure level (P). “**NNN**” indicates the accumulation or level value. The value listed above indicates that accumulated precipitation (GRIB code 61) with a 3-hourly accumulation interval should be verified. 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.

- `0` indicates no additional processing is to be done.
- `1` indicates missing data in the observation field should be used to mask the forecast field.

- `2` indicates missing data in the forecast field should be used to mask the observation field.
- `3` indicates masking should be performed in both directions (i.e., mask the forecast field with the observation field and vice-versa).

Prior to defining objects, it is recommended that the raw fields be made to look similar to each other by assigning a value of 3 to this parameter. 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 the set of pre-defined masks included with the MODE tool 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 that 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. Users can easily define and apply new masking polygons similar to the predefined ones. 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 that the raw fields should be made to look similar to each other. For example, if the model only predicts values for a variable above some threshold, the observations should be thresholded at that same level. The thresholds are specified using the Fortran conventions of `gt`, `ge`, `eq`, `ne`, `lt`, `le` to indicate 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 with an intensity percentile that 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 the forecast field.
- `3` indicates that no additional merging is allowed in either field, meaning that each object will match at most one object in the other field.

By default, additional merging is allowed in both fields.

```
max_centroid_dist = 800/grid_res;
```

Computing the attributes for all possible pairs of objects can take some time depending on the numbers of objects. The **`max_centroid_dist`** variable is used to specify how far apart objects should be in order to conclude that they have no chance of matching. No pairwise attributes are computed for pairs of objects whose centroids are farther away than this distance, defined in terms of grid units. Setting this 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 weighted sum is 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 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 details. 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 that have an aspect ratio that 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. Two objects that are very different in size will have an area ratio that is close to zero. For cases like this, the **`area_ratio_conf`** variable allows the user to assign less confidence to the measured 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 that have an interest value that is above this threshold will be matched, while those with an interest value that is below this threshold will remain unmatched. Increasing the threshold will decrease the number of matches while decreasing the threshold will increase the number of matches. By default, the total interest threshold is set to 0.7.

```
print_interest_threshold = 0.0;
```

The **print_interest_threshold** variable determines which pairs of object attributes will be written to the output object attribute ASCII file. The user may choose to set the **print_interest_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.

```
met_data_dir = "MET_BASE/data";
```

The MODE tool uses several static data files when generating plots. If it cannot find the data it needs in the expected directory, it will error out. The **met_data_dir** variable may be set to the path that contains the `data/` subdirectory of the top-level MET directory to instruct MODE where to find the static data files.

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

The **fcst_raw_color_table, obs_raw_color_table** and **mode_color_table** variables indicate which color table files are to be used when generating the output PostScript plot. The forecast and observation raw field are plotted using values in the **fcst_raw_color_table** and **obs_raw_color_table**, while the objects 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.

```
fcst_raw_plot_min = 0.0;
fcst_raw_plot_max = 0.0;
obs_raw_plot_min  = 0.0;
obs_raw_plot_max  = 0.0;
```

These variables indicate the min and max data values to be plotted for the forecast and observation fields. If set to non-zero values, the forecast and observation raw color tables specified above will be rescaled to match the specified range.

```
stride_length = 1;
```

The MODE tool generates a colorbar to represent the contents of the colortable that was used to plot a field of data. The number of entries in the colorbar matches the number of entries in the colortable. The values defined for each color in the colortable are also plotted next to the colorbar. The **`stride_length`** variable is used to define the frequency with which the colortable values should be plotted. Setting this variable to 1, as shown above, indicates that every colortable value should be plotted. Setting it to an integer, n > 1, indicates that only every nth colortable value should be plotted.

```
zero_border_size = 1;
```

The MODE tool is not able to define objects that 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.

```
plot_valid_flag = 0;
```

When applied, the **`plot_valid_flag`** variable indicates that only the region containing valid data after masking is applied should be plotted.

- **`0`** indicates the entire domain should be plotted.
- **`1`** indicates only the region containing valid data after masking should be plotted.

The default value of this flag is 0.

```
plot_gcarc_flag = 0;
```

When applied, the **`plot_gcarc_flag`** variable indicates that the edges of polylines should be plotted using great circle arcs as opposed to straight lines in the grid. The default value of this flag is 0.

```
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.

```
version = "V1.1";
```

The **`version`** indicates the version of the `mode` configuration file used. Future versions of MET may include changes to `mode` and the `mode` configuration file. This value should not be modified.

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 containing column names. The second line contains data comparing the two raw fields after any masking of bad data or based on a grid or lat/lon polygon has been applied. The third contains data comparing the two fields after any raw thresholds have been applied. The fourth, and last line, contains data comparing the derived object fields scored using traditional measures. This file uses the following naming convention: **`mode_`FCST_VAR_LVL`_vs_` OBS_VAR_LVL_HHMMSS`L`_YYYYMMDD_HHMMSS`V`_ HHMMSS`A_cts.txt`** where **FCST_VAR_LVL** is the forecast variable and vertical level being used, **OBS_VAR_LVL** is the observation variable and vertical level being used, **HHMMSS`L`** indicates the forecast lead time, **YYYYMMDD_HHMMSS`V`** indicates the forecast valid time, and **HHMMSS`A`** 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. This CTS output file differs somewhat from the CTS output of the Point-Stat and Grid-Stat tools. The columns of this output file are summarized in table 6-1.

Table 6-1. *Format of CTS output file.*

MODE ASCII CONTINGENCY TABLE OUTPUT FORMAT		
Column Number	**MODE CTS Column Name**	**Description**
1	`MODEL`	Model identifier
2	`FCST_LEAD`	Forecast lead time (in HHMMSS format)
3	`FCST_VALID`	Forecast valid time (in YYYYMMDD_HHMMSS format)
4	`FCST_ACCUM`	Forecast length of accumulation period (in HHMMSS format; for precipitation only)
5	`LEAD2`	"Lead time" (in HHMMSS) for Field2, which is usually an observed field observation; when field 2 is an observed field this should be "000000"
6	`OBS_VALID`	Observation valid time (in YYYYMMDD_HHMMSS format)

MODE ASCII CONTINGENCY TABLE OUTPUT FORMAT		
Column Number	**MODE CTS Column Name**	**Description**
7	`OBS_ACCUM`	Observation length of accumulation period (in HHMMSS format; for precipitation only)
8	`FCST_RAD`	Forecast convolution radius (in grid squares)
9	`FCST_THR`	Forecast convolution threshold (various units)
10	`OBS_RAD`	Observation convolution radius (in grid squares)
11	`OBS_THR`	Observation convolution threshold (various units)
12	`FCST_VAR`	Forecast variable being verified
13	`FCST_LEV`	Forecast vertical level at which verification performed
14	`OBS_VAR`	Observation variable being verified
15	`OBS_LEV`	Observation vertical level at which verification performed
16	`FIELD`	Field over which the contingency table was computed (**RAW**, **FILTER**, or **OBJECT**)
17	`=`	Equal sign delimiter
18	`TOTAL`	Total number of matched pairs
19	`FY_OY`	Number of forecast yes and observation yes
20	`FY_ON`	Number of forecast yes and observation no
21	`FN_OY`	Number of forecast no and observation yes
22	`FN_ON`	Number of forecast no and observation no
23	`BASER`	Base rate
24	`FMEAN`	Forecast mean
25	`ACC`	Accuracy
26	`FBIAS`	Frequency Bias
27	`PODY`	Probability of detecting yes
28	`PODN`	Probability of detecting no
29	`POFD`	Probability of false detection
30	`FAR`	False alarm ratio
31	`CSI`	Critical Success Index
32	`GSS`	Gilbert Skill Score
33	`HK`	Hanssen-Kuipers Discriminant
34	`HSS`	Heidke Skill Score
35	`ODDS`	Odds Ratio

The second ASCII file the MODE tool generates contains all of the attributes for simple 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 first row of every MODE object attribute file is a header containing a column names. 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: **`F`NNN**, **`O`NNN**,

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 1-38 and fill data elsewhere. The next form corresponds to attributes for pairs of simple forecast and observation objects. These rows contain valid data in columns 1-18 and 39-50 and fill data elsewhere. The next two forms correspond to attributes for merged composite forecast and observation objects. These rows contain valid data in columns 1-38 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 1-18 and 39-49 and fill data elsewhere. These object identifiers are described in Table 6-2. The generation of this file can be disabled using the **-obj_stat** command line option.

Table 6-2. *Object identifier descriptions for MODE object attribute output files.*

Object identifier (object_id)	Valid Data Columns	Description of valid data
FNNN, ONNN	1-38	Attributes for simple forecast, observation objects
FNNN_ONNN	1-18, 39-50	Attributes for pairs of simple forecast and observation objects
CFNNN, CONNN	1-38	Attributes for merged composite objects in forecast, observation fields
CFNNN_CONNN	1-18, 39-49	Attributes for pairs of forecast, observation composite objects

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 Table 6-3.

Table 6-3. *Format of MODE object attribute output files.*

MODE ASCII OBJECT ATTRIBUTE OUTPUT FORMAT		
Column Number	**MODE Column Name**	**Description**
1	MODEL	Model identifier
2	FCST_LEAD	Forecast lead time (in HHMMSS format)
3	FCST_VALID	Forecast valid time (in YYYYMMDD_HHMMSS format)

MODE ASCII OBJECT ATTRIBUTE OUTPUT FORMAT		
Column Number	**MODE Column Name**	**Description**
4	FCST_ACCUM	Forecast length of accumulation period (in HHMMSS format; for precipitation only)
5	LEAD2	"Lead time" (in HHMMSS) for Field2, which is usually an observed field observation; when field 2 is an observed field this should be "000000"
6	OBS_VALID	Observation valid time (in YYYYMMDD_HHMMSS format)
7	OBS_ACCUM	Observation length of accumulation period (in HHMMSS format; for precipitation only)
8	FCST_RAD	Forecast convolution radius (in grid squares)
9	FCST_THR	Forecast convolution threshold (various units)
10	OBS_RAD	Observation convolution radius (in grid squares)
11	OBS_THR	Observation convolution threshold (various units)
12	FCST_VAR	Forecast variable being verified
13	FCST_LEV	Forecast vertical level at which verification performed
14	OBS_VAR	Observation variable being verified
15	OBS_LEV	Observation vertical level at which verification performed
16	OBJECT_ID	Object numbered from 1 to the number of objects in each field
17	OBJECT_CAT	Object category indicating to which composite object it belongs
18	=	Equal sign delimiter
19-20	CENTROID_X, _Y	Location of the centroid (in grid units)
21-22	CENTROID_LAT, _LON	Location of the centroid (in lat/lon degrees)
23	AXIS_ANG	Object axis angle (in degrees)
24	LENGTH	Length of the enclosing rectangle (in grid units)
25	WIDTH	Width of the enclosing rectangle (in grid units)
26	AREA	Object area (in grid squares)
27	AREA_FILTER	Area of the object containing non-zero data in the filtered field (in grid squares)
28	CURVATURE	Radius of curvature of the object defined in terms of third order moments (in grid units)
29-30	CURVATURE_X, _Y	Center of curvature (in grid coordinates)
31	COMPLEXITY	Ratio of the area of an object to the area of its convex hull (unitless)
32-36	INTENSITY_10, _25, _50, _75, _90	10th, 25th, 50th, 75th, and 90th percentiles of intensity of the filtered field within the object (various units)

MODE ASCII OBJECT ATTRIBUTE OUTPUT FORMAT		
Column Number	**MODE Column Name**	**Description**
37	INTENSITY_NN	The percentile of intensity chosen for use in the percentile intensity ratio (column 49; variable units)
38	INTENSITY_SUM	Sum of the intensities of the filtered field within the object (variable units)
39	CENTROID_DIST	Distance between two objects centroids (in grid units)
40	BOUNDARY_DIST	Minimum distance between the boundaries of two objects (in grid units)
41	CONVEX_HULL_DIST	Minimum distance between the convex hulls of two objects (in grid units)
42	ANGLE_DIFF	Difference between the axis angles of two objects (in degrees)
43	AREA_RATIO	Ratio of the areas of two objects defined as the lesser of the forecast area divided by the observation area or its reciprocal (unitless)
44	INTERSECTION_AREA	Intersection area of two objects (in grid squares)
45	UNION_AREA	Union area of two objects (in grid squares)
46	SYMMETRIC_DIFF	Symmetric difference of two objects (in grid squares)
47	INTERSECTION_OVER_AREA	Ratio of intersection area to union area (unitless)
48	COMPLEXITY_RATIO	Ratio of complexities of two objects defined as the lesser of the forecast complexity divided by the observation complexity or its reciprocal (unitless)
49	PERCENTILE_INTENSITY_RATIO	Ratio of the nth percentile (column 37) of intensity of the two objects defined as the lesser of the forecast intensity divided by the observation intensity or its reciprocal (unitless)
50	INTEREST	Total interest value computed for a pair of simple objects (unitless)

The MODE tool creates a NetCDF output file containing the object fields that are 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 Tables 6-4 and 6-5.

Table 6-4. NetCDF dimensions for MODE output.

`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)

Table 6-5. Variables contained in MODE NetCDF output.

`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. Examples of the PostScript plots can be obtained by running the example cases provided with the MET tarball.

The first page of PostScript output contains a great deal of summary information. Six tiles of images provide thumbnail images of the raw fields, matched/merged object fields, and object index fields for the forecast and observation grids. In the matched/merged object fields, matching colors of objects across fields indicate that the corresponding objects match, while within a single field, black outlines indicate merging. Note that objects that are colored royal blue are unmatched. Along the bottom of the page, the criteria used for object definition and matching/merging are listed. Along the right side of the page, total interest values for pairs of simple objects are listed in sorted order. The numbers in this list correspond to the object indices shown in the object index plots.

The second and third pages of the PostScript output file display enlargements of the forecast and observation raw and object fields, respectively. The fourth page displays the forecast object with the outlines of the observation objects overlaid, and vice versa.

If the double threshold merging or the fuzzy engine merging techniques have been applied, the output from those steps is summarized on additional pages.

Chapter 7 – The VSDB-Analysis Tool

7.1 Introduction

The VSDB-Analysis tool ties together results from the Point-Stat and Grid-Stat tools by providing summary statistical information and a way to filter their VSDB output files.

The VSDB-Analysis tool requires VSDB output from Point-Stat and/or Grid-Stat. See Sections 4.3.3 and 5.3.3, respectively, for information on the VSDB output format of the Point-Stat and Grid-Stat tools.

7.2 Scientific and statistical aspects

The VSDB-Analysis tool (i) aggregates results over a user-specified time; (ii) stratifies statistics based on time of day, model initialization time, lead-time, model run identifier, or output filename; and (iii) computes specific verification indices such as the GO Index[1]. Future functionality may include information about time-trends and/or calculations based on climatology (e.g., anomaly correlation). This section summarizes the capabilities of the VSDB-Analysis tool and describes how the GO Index, summary statistics, and aggregated statistics are computed.

7.2.1 Filter VSDB lines

The VSDB-Analysis tool can be used to simply filter out specific VSDB lines based on user-specified search criteria. All of the VSDB lines that are retained from one or many files are written to a single output file.

7.2.2 Summary statistics for columns

The VSDB-Analysis tool can be used to produce summary information for a single column of data. After the user specifies the specific line type, specific column of interest, and any other relevant search criteria, summary information is produced from values in that column of data. The summary statistics produced are: mean, standard deviation, minimum, maximum, and the 10th, 25th, 50th, 75th, and 90th percentiles.

Confidence intervals are computed for the mean and standard deviation of the column of data. For the mean, the confidence interval is computed two ways – based on an assumption of normality and also using the bootstrap method. For the standard deviation, the confidence interval is computed using the bootstrap method. In this application of the bootstrap method, the values in the column of data being summarized are resampled, and for each replicated sample, the mean and standard deviation are computed.

[1] The GO Index is a summary measure for NWP models that is used by the Air Force Weather Agency. It combines verification statistics for several forecast variables and lead times.

7.2.3 Aggregated values from multiple VSDB lines

The VSDB-Analysis tool can be used to create aggregated values from multiple VSDB lines of the same type. The user may specify the specific line type of interest and any other relevant search criteria. The VSDB-Analysis tool then creates sums of each of the values in all lines matching the search criteria. The aggregated data are output as the same line type as the user specified. The VSDB line types which may be aggregated in this way are the contingency table (FHO, CTC, CTP, CFP, and COP) and partial sums (SL1L2, SAL1L2, VL1L2, and VAL1L2) line types.

7.2.4 Aggregate VSDB lines and produce aggregated statistics

The VSDB-Analysis tool can be used to aggregate multiple VSDB lines of the same type together and produce relevant statistics from the aggregated line. This may be done in the same manner listed above in 7.2.3. However, rather than writing out the aggregated VSDB line itself, the relevant statistics generated from that aggregated line are provided in the output. Specifically, if a contingency table line type (FHO, CTC, CTP, CFP, or COP) has been aggregated, a contingency table statistics (CTS) line type will be written out. If a partial sums line type (SL1L2 or SAL1L2) has been aggregated, a continuous statistics (CNT) line type will be written out. If the matched pair line type (MPR) has been aggregated, the user may choose the line type to be output (FHO, CTC, CTP, CFP, COP, CTS, CNT, SL1L2 or SAL1L2). Note that no relevant statistics are produced for the vector partial sums line types (VL1L2 or VAL1L2).

When aggregating the matched pair line type (MPR) and computing an output contingency table statistics (CTS) or continuous statistics (CNT) line type, the bootstrapping method is applied for computing confidence intervals. The bootstrapping method is applied here in the same way that it is applied in the Point-Stat and Grid-Stat tools. For a set of n matched forecast-observation pairs, the matched pairs are resampled with replacement many times. For each replicated sample, the corresponding statitistics are computed. The confidence intervals are derived from the statistics computed for each replicated sample.

7.2.5 GO Index

The VSDB-Analysis tool can be used to calculate the GO Index. This measure is a weighted average of the RMSE values for wind speed, dewpoint temperature, temperature, height, and pressure at several levels in the atmosphere. The variables, levels, and lead times included in the index are shown in Table 7-1. The partial sums (SL1L2 lines in the VSDB output) for each of these variables at each level and lead time must have been computed in a previous step. The MET analysis tool then uses the weights in Table 7-1 to compute values for the GO Index.

Table 7-1. Variables, levels, and weights used to compute the GO Index.

Variable	Level	Weights by leadtime			
		12 h	24 h	36 h	48 h
Wind speed	250 hPa	4	3	2	1
	400 hPa	4	3	2	1
	850 hPa	4	3	2	1
	Surface	8	6	4	2
Dewpoint temperature	400 hPa	8	6	4	2
	700 hPa	8	6	4	2
	850 hPa	8	6	4	2
	Surface	8	6	4	2
Temperature	400 hPa	4	3	2	1
	Surface	8	6	4	2
Height	400 hPa	4	3	2	1
Pressure	Mean sea level	8	6	4	2

7.3 Practical information

The following sections describe the usage statement, required arguments and optional arguments for the Vsdb-Analysis tool.

7.3.1 `vsdb_analysis` *usage*

The usage statement for the VSDB-Analysis tool is shown below:

```
Usage: vsdb_analysis
        -lookin path
        [-out name]
        [-v level]
        [-config config_file] | [JOB COMMAND LINE]
```

`vsdb_analysis` has two required arguments and accepts up to three optional ones.

In the usage statement for the VSDB analysis tool, some additional terminology is introduced. In the VSDB analysis tool, the term “job” refers to a set of tasks to be performed after applying user-specified options (i.e., “filters”). The filters are used to pare down a collection of output from `grid_stat` and/or `point_stat` to only those lines that are desired for the analysis. The job and its filters together comprise the “job command line”. The “job command line” may be specified either on the command line to run a single analysis job or within the configuration file to run multiple analysis jobs at the same time. If jobs are specified in both the configuration file and the command line, only the jobs indicated in the configuration file will be run. The various jobs types are described in Table 7-2 and the filtering options are described in section 7.3.2.

Required arguments for `vsdb_analysis`

1. The `–lookin` **path** specifies the name of a specific VSDB file (any file ending in .vsdb) or the name of a directory where the VSDB-Analysis tool will search for VSDB files. This option may be used multiple times to specify multiple locations.
2. Either a configuration file must be specified with the `–config` option, or a "`JOB COMMAND LINE`" must be denoted. The "`JOB COMMAND LINE`" is described in section 7.3.2,

Optional arguments for `vsdb_analysis`

1. The `–config` **config_file** specifies the configuration file to be used. The contents of the configuration file are discussed below.
2. The `–out` **name** option indicates the filename to which output data should be written. If this option is not used, the output is directed to standard output.
3. The `–v` **level** 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 `vsdb_analysis` calling sequence is shown below.

```
vsdb_analysis  -lookin ../out/point_stat
               -config config/VSDBAnalysisConfig
```

In this example, the vsdb analysis tool will search for valid VSDB lines located in the `../out/point_stat` directory that meet the options specified in the configuration file, `config/VSDBAnalysisConfig`.

7.3.2 `vsdb_analysis` configuration file

The default configuration file for the vsdb analysis tool named `VSDBAnalysisConfig_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`. Like the other configuration files described in this document, it is recommended that users make a copy of these files prior to modifying their contents.

Most of the user-specified parameters listed in the VSDB analysis configuration file are used to filter the ASCII statistical output from `grid_stat` and/or `point_stat` down to a desired subset of lines over which statistics are to be computed. Only output from grid_stat and/or point_stat that meet all of the parameters specified in the VSDB analysis configuration file will be retained.

The configuration file for the VSDB analysis tool contains many comments describing its contents. A more detailed explanation of the user-specified parameters is provided below.

Note that the options specified in the first section of the configuration file below, prior to the joblist, will be applied to every job specified in the joblist. However, if an individual job specifies a particular option that was specified above, it will be applied for that job. For example, if the model[] option is set at the top to [“Run 1”, “Run2”], but a job in the joblist sets the –model option as “Run1”, that job will be performed only on “Run1” data.

Also note that environment variables may be used when editing configuration files, as described in the section 3.5.2 for the PB2NC.

```
model[] = [];
```

This option allows the user to filter according to column 2 of the grid_stat/point_stat VSDB output. The model[] option allows the user to specify a comma-separated list of model names to be used for any analyses to be performed. The names must be in double quotation marks. If multiple models are listed, the analyses will be performed on their union. These selections may be further refined by using the “-model” option within the job command lines.

```
fcst_lead[] = [];
```

The fcst_lead[] option filters according to column 3 of the grid_stat/point_stat VSDB output. The user may specify a comma-separated list of forecast times in HH[MMSS] format to be used for any analyses to be performed. If multiple forecast times are listed, the analyses will be performed on their union. These selections may be further refined by using the "-fcst_lead" option within the job command lines.

```
valid_begin = "";
valid_end   = "";
```

These two options allow the user to stratify by column 4 of the grid_stat/point_stat VSDB output. One may specify the beginning and ending valid times in YYYYMMDD[_HH[MMSS]] format to be used for all analyses performed. If multiple valid times fall within the valid time window, the analyses will be performed on their union. These selections may be further refined by using the "-valid_begin" and "-valid_end" options within the job command line.

```
init_begin = "";
init_end   = "";
```

These options allow the user to stratify by model initialization time, derived from columns 3 and 4 of the `grid_stat/point_stat` VSDB output. These specify the beginning and ending model initialization times in YYYYMMDD[_HH[MMSS]] format to be used for all analyses performed. If multiple model initialization times fall within the initialization time window the, analyses will be performed on their union. These selections may be further refined by using the `"-init_begin"` and `"-init_end"` options within the job command line.

```
init_hour[] = [""];
```

This option allows the user to specify a comma separated list of model initialization hours to be filtered out of the data. The initialization hour is formatted as HH[MMSS] and may be used to aggregate data over multiple runs all initialized at the same time of day.

```
obtype[] = [];
```

The `obtype[]` option allows the user to stratify by the observation types to be used for all analyses (i.e. column 5 of the `grid_stat/point_stat` VSDB output). If multiple observation types are listed, the analyses will be performed on their union. These selections may be further refined by using the `"-obtype"` option within the job command line.

```
vx_mask[] = [ ];
```

This option stratifies based on masking regions to be used for all analyses performed (column 6 of the `grid_stat/point_stat` VSDB output). If multiple verification masking regions are listed, the analyses will be performed on their union. These selections may be further refined by using the `"-vx_mask"` option within the job command line.

```
line_type[] = [];
```

The `line_type[]` option stratifies column 7 of of the `grid_stat/point_stat` VSDB output based on a comma-separated list of regular expressions for all analyses to be performed. If multiple regular expressions for Grid Stat/Point Stat line types are listed, the analyses will be performed on their union. These selections may be further refined by using the `"-line_type"` option within the job command line. An example of valid options follows:

```
line_type[] = [ "FHO>=0.0000", "CNT/0.050", "SL1L2" ];
```

This selection of options would search output from Grid Stat/Point Stat for lines that contain FHO statistics for a threshold greater than or equal to 0.000, contain confidence intervals for continuous verification statistics (CNT) with an alpha value of 0.05, or contain SL1L2 type statistics.

`var[] = [];`

The `var[]` option allows the user to provide a list of model variables to be used for all analyses to be performed (column 8 of the `grid_stat`/`point_stat` VSDB output). If multiple model variables are listed, the analyses will be performed on their union. These selections may be further refined by using the "`-var`" option within the job command line.

`level[] = [];`

This option allows the user to provide a comma-separated list of model levels to be used for all analyses performed (column 9 of the `grid_stat`/`point_stat` VSDB output). Levels in the list may refer to a single level (e.g. "SFC", "P500") or a range of levels (e.g. "P750-950"). If multiple model levels are listed, the analyses will be performed on their union. These selections may be further refined by using the "`-level`" option within the job command line.

`interp_mthd[] = [];`

This option stratifies based on a comma-separated list of interpolation methods to be used for all analyses performed (the second-to-last column of the `grid_stat`/`point_stat` VSDB output). If multiple interpolation methods are listed, the analyses will be performed on their union. These selections may be further refined by using the "`-interp_mthd`" option within the job command line.

`interp_pnts[] = [];`

This option stratifies based on a comma-separated list of interpolation points to be used for all analyses performed (the last column of the `grid_stat`/`point_stat` VSDB output). If multiple interpolation points are listed, the analyses will be performed on their union. These selections may be further refined by using the "`-interp_pnts`" option within the job command line.

```
joblist[] = [
"-job vsdb_job_filter -dump_row ./filter_job.vsdb"
];
```

The `joblist[]` refers to the job command (the analysis job and filter options) to be performed. Each entry in the joblist contains the task and options for a single analysis to be performed. The format for an analysis job is as follows:

`-job job_name` OPTIONAL ARGS

All possible tasks for `job_name` are listed in Table 7-2.

Table 7-2. Description of components of the job command lines for the VSDB analysis tool.

	Description	Required Arguments
`vsdb_job_filter`	Filters out the lines output from Grid Stat/Point Stat based on applying options	`-dump_row`
`vsdb_job_summary`	Computes the mean, standard deviation, and percentiles (min, 10th, 25th, 50th, 75th, 90th, and max)	`-line_type` `-column`
`vsdb_job_aggr`	Aggregates the Grid Stat/Point Stat output, computing the statistic specified for the entire collection of valid line	`-line_type` (may be set to FHO, CTC, CTP, CFP, COP, SL1L2, SAL1L2, VL1L2, or VAL1L2)
`vsdb_job_aggr_cts`	Aggregates contingency table type lines and produces aggregated contingency table statistics	`-line_type` (may be set to FHO, CTC, CTP, CFP, or COP)
`vsdb_job_aggr_cnt`	Aggregates SL1L2 or SAL1L2 lines and produces statistics dervides from the aggregated partial sums	`-line_type` (may be set to SL1L2 or SAL1L2)

`vsdb_job_aggr_mpr`	Aggregates MPR lines and produces statistics derived from the aggregated matched pairs	`-out_line_type`
		(may be set to FHO, CTC, CTP, CFP, COP, CTS, CNT, SL1L2 or SAL1L2)
		`-threshold (for out_line type of FHO, CTC, CTP, CFP, COP, or CTS)`

`vsdb_job_go_index`	Calculates the GO Index as described in section 7.1.1.	`-init_begin` `-init_end`

In addition to the job-specific options that may be invoked, the following options may also be used on the job command line:

- "`-alpha` value" to select the alpha value for the analysis. This parameter is the same as described for the configuration file for Point-Stat (refer to Chapter 4).
- "`-model` name" to select the model name for the analysis.
- "`-fcst_lead` HH[MMSS]" to select the forecast time for the analysis.
- "`-valid_begin` YYYYMMDD[_HH[MMSS]]" to select the beginning of the valid time window for the analysis (may be used only once).
- "`-valid_end` YYYYMMDD[_HH[MMSS]]" to select the ending of the valid time window for the analysis (may be used only once).
- "`-init_begin` YYYYMMDD[_HH[MMSS]]" to select the beginning of the initialization time window for the analysis (may be used only once).
- "`-init_end` YYYYMMDD[_HH[MMSS]]" to select the ending of the initialization time window for the analysis (may be used only once).
- "`-init_hour` HH[MMSS]" to select the initialization hour for the analysis.
- "`-obtype` name" to select the observation type for the analysis.
- "`-vx_mask` name" to select the verification masking region for the analysis.
- "`-line_type` name" to select the VSDB line type for the analysis.
- "`-column` name" to select the column of the VSDB line type for the analysis.
- "`-var` name" to select the variable for the analysis.
- "`-level` name" to select the vertical level for the analysis.
- "`-interp_mthd` name" to select the interpolation method for the analysis.
- "`-interp_pnts` n" to select the number of interpolation points for the analysis.
- "`-dump_row` path" to select the filename to which the VSDB lines used for the analysis should be written.
- "`-boot_interval` value" to override the default bootstrap interval method (may be used only once).
- "`-boot_rep_`prop value" to override the default bootstrap replicate proportion size (may be used only once).
- "`-n_boot_rep` value" to override the default number of bootstrap replicates to be used (may be used only once).
- "`-boot_rng` value" to override the default bootstrap random number generator to be used (may be used only once).
- "`-boot_seed` value" to override the default bootstrap random number generator seed value to be used (may be used only once).

These options are the same as described in other parts of the VSDB analysis tool configuration file.

```
boot_interval = 1;
```

The **`boot_interval`** variable indicates what method should be used for computing bootstrap confidence intervals. A value of 0 indicates that the highly accurate but computationally intensive BCa (bias-corrected percentile) method should be used. A value of 1 indicates that the somewhat less accurate but efficient percentile method should be used.

```
boot_rep_prop = 1.0;
```

The **`boot_rep_prop`** variable must be set to a value between 0 and 1. When computing bootstrap confidence intervals over n sets of matched pairs, the size of the subsample, m, may be chosen less than or equal to the size of the sample, n. This variable defines the size of m as a proportion relative to the size of n. A value of 1, as shown above, indicates that the size of the subsample, m, should be equal to the size of the sample, n.

```
n_boot_rep = 1000;
```

The **`n_boot_rep`** variable defines the number of subsamples that should be taken when computing bootstrap confidence intervals. This variable should be set large enough so that when confidence intervals are computed multiple times for the same set of data, the intervals do not change much. Setting this variable to zero disables the computation of bootstrap confidence intervals which may be necessary to run in realtime or near-realtime over large domains. Setting this variable to 1000, as shown above, indicates that bootstrap confidence interval should be computed over 1000 subsamples of the matched pairs.

```
boot_rng = "mt19937";
```

The **`boot_rng`** variable defines the random number generator to be used in the computation of bootstrap confidence intervals. Subsamples are chosen at random from the full set of matched pairs. The randomness is determined by the random number generator specified. Users should refer to detailed documentation of the GNU Scientific Library for a listing of the random number generators available for use.

```
boot_seed = “”;
```

The **`boot_seed`** variable may be set to a specific value to make the computation of bootstrap confidence intervals fully repeatable. When left empty, as show above, the random number generator seed is chosen automatically which will lead to slightly different bootstrap confidence intervals being computed each time the data is run. Specifying a value here ensures that the bootstrap confidence intervals will be computed the same over multiple runs of the same data.

```
rank_corr_flag = 1;
```

The **`rank_corr_flag`** variable may be set to 0 ("no") or 1 ("yes") to indicate whether or not Kendall's Tau and Spearman's Rank correlation coefficients should be computed. The computation of these rank correlation coefficients is very slow when run over many matched pairs. By default, this flag is turned on, as shown above, but setting it to 0 should improve the runtime performance.

```
tmp_dir = "/tmp";
```

This parameter indicates the directory where the vsdb analysis tool should write temporary files.

```
version = "V1.1";
```

`version` indicates the version number for the contents of this configuration file. The value should generally not be modified.

7.3.3 VSDB-Analysis tool output

The output generated by the VSDB Analysis tool contains statistics produced by the analysis. It also records information about the analysis job that produced the output for each line. The format of output from each VSDB job command is described below.

vsdb_job_filter

This job command finds and filters VSDB lines down to those meeting criteria specified by the filter's options. The output is written to a file specified by the "-dump_row" option.

The output of this job is the same VSDB format described in section 4.3.3 of Chapter 4.

vsdb_job_summary

This job produces summary statistics for the column name and line type specified by the "`-column`" and "`-line_type`" options. The output of this job type consists of three lines. The first line contains "`COL_NAME`", followed by a colon, then the column names for the data in the next line. The second line contains the word "`SUMMARY`" in the first column, followed by a colon, then the total, mean with confidence intervals, standard deviation with confidence intervals, minimum value, percentiles (10th, 25th, 50th, 75th, and 90th) and the maximum value. The third line contains "`JOB_COMMAND`" followed by a colon, then the filtering parameters used for this job. The output column are shown in Table 7-3 below.

Table 7-3. Columnar output of "`vsdb_job_summary`" output from the VSDB-Analysis tool.

Column Number	Description
1	`SUMMARY:` (job type)
2	Total
3-7	Mean including normal and bootstrap upper and lower confidence limits
8-10	Standard deviation including bootstrap upper and lower confidence limits
11	Minimum
12	10th percentile
13	25th percentile
14	Median (50th percentile)
15	75th percentile
16	90th percentile
17	Maximum value

vsdb_job_aggr, vsdb_job_aggr_cts, vsdb_job_aggr_cnt

These jobs aggregate output from the VSDB line type specified using the "`-line_type`" argument. The output of this job type is in the same format as the line type specified (see Section 4.3.3), except that the header only consists of the name of the line type and an equal sign. Also, the interpolation method and interpolation widths are not output, but instead the job command used is recorded in parentheses.

vsdb_job_aggr_mpr

This job aggregates the matched pair data from the MPR line type and may be used to compute whichever line type is requested using the "`-out_line_type`" argument. For contingency table line types, the "`-threshold`" arguement must also be supplied to incidcate the desire type of thresholding to be applied. Generally, aggregating the matched pair values themselves across multiple cases is unrealistic due to storage limitations. However, relatively small collections of matched pairs, this may be reasonable to do.

vsdb_job_go_index

The output from this job type consists of the words "GO_INDEX" followed by a colon and then the value of the GO Index. The job command used to produce the value is listed on the following line of output.

7.4 Future additions

In addition to providing statistical summaries, in the future these analysis tools will also have accompanying scripts/subroutines supporting 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 users an easy way to obtain overviews of the statistics produced by the Point-Stat, Grid-Stat, and the VSDB-Analysis tools. They can also be used for comparison to graphics produced by other graphics packages (e.g., to ensure the data are being plotted correctly). 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 8 – The MODE-Analysis Tool

8.1 Introduction

MODE output files can be quite large; currently, these files contain 50 columns. This means that it is very difficult – effectively impossible – to interpret the results by simply browsing the files. Furthermore, for particular applications some data fields in the MODE output files may not be of interest. Because of these and similar considerations, the MET development team believed a tool providing basic summary statistics and filtering capabilities for these files would be helpful for many users. The MODE analysis tool provides these capabilities. Users who are not proficient at writing scripts can use the tool directly, and even those using their own scripts can use this tool as a filter, to extract only the MODE output lines that are relevant for their application.

8.2 Scientific and statistical aspects

The MODE-Analysis tool operates in two modes, called "summary" and "by case". In summary mode, the user provides (on the command line or via a configuration file) information regarding which fields (columns) are of interest, as well as matching criteria that determine which lines in each file are used and which are ignored. For example, a user may be interested in forecast object areas, but only if the object was matched, and only if the object centroid is inside a particular region. The summary statistics generated (for each field) are the minimum, maximum, mean, standard deviation, and the10th, 25th, 50th, 75th and 90th percentiles. In addition, the user may specify a "dump" file: the individual MODE lines used to produce the statistics will be written to this file. This option provides the user with a filtering capability. The dump file will consist only of lines that match the specified criteria.

The other mode for operating the analysis tool is "by case". Given initial and final values for forecast lead time, the tool will output, for each valid time in the interval, the matched area, unmatched area, and the number of forecast and observed objects that were matched or unmatched. For the areas, the user can specify forecast or observed objects, and also simple or composite objects. A dump file may also be specified in this mode.

8.3 Practical information

The MODE-Analysis tool reads lines from MODE output files and applies filtering and computes basic statistics on the object attribute values. For each job type, filter parameters can be set to determine which MODE output lines are used. The following sections describe the `mode_analysis` usage statement, required arguments, and optional arguments.

8.3.1 `mode_analysis` *usage*

The usage statement for the MODE-Analysis tool is shown below:

```
Usage: mode_analysis
        -lookin path
        -summary | -bycase
        [-column name]
        [-dump_row filename]
        [-out filename]
        [-help]
        [MODE FILE LIST]
        [-config config_file] | [MODE LINE OPTIONS]
```

Required arguments for `mode_analysis`

The MODE-Analysis tool requires specification of a "job type" and a filename or directory indicated by the `-lookin` option. The `–lookin` option may be called multiple times.

The MODE-Analysis tool can perform two basic types of jobs, which are identified as follows:

`-summary`
`-bycase`

Exactly one of these job types must be specified.

Specifying "`-summary`" will produce summary statistics for the MODE output column specified. For this job type, a column name (or column number) must be specified using the "`-column`" option. Column names are not case sensitive. The column names are the same as described in Section 6.3.3 of Chapter 6. More information about this option is provided in subsequent sections.

Specifying "`-bycase`" will produce a table of metrics for each case undergoing analysis. Any columns specified are ignored for this option.

The output for each of these jobs types is described in later sections.

Optional arguments for `mode_analysis`

The `mode_analysis` options are described in the following section. These are divided into sub-sections describing the analysis options and mode line options.

Analysis options

The general analysis options described below provide a way for the user to indicate configuration files to be used, where to write lines used to perform the analysis, and over which fields to generate statistics.

`-config` filename

This option gives the name of a configuration file to be read. The contents of the configuration file are described in Section 8.3.3.

`-dump_row` filename

Any MODE lines kept from the input files are written to **filename**.

`-column` column

Specifies which columns in the MODE output files to generate statistics for. Fields may be indicated by name (case insensitive) or column number (beginning at one). This option can be repeated as often as needed to specify multiple columns.

MODE line options

MODE line options are used to create filters that determine which of the MODE output lines that are read in, are kept. The MODE line options are numerous. They fall into seven categories: toggles, multiple set string options, multiple set integer options, integer max/min options, date/time max/min options, floating-point max/min options, and miscellaneous options. These options are described in subsequent sections.

Toggles

The MODE line options described in this section are shown in pairs. These toggles represent parameters that can have only one (or none) of two values. Any of these toggles may be left unspecified. However, if neither option for each toggle is indicated, the analysis will produce results that combine data from both toggles. This may produce unintended results.

`-fcst / -obs`

This toggle indicates whether forecast or observed lines should be used for analysis.

`-single` / `-pair`

This toggle indicates whether single object or object pair lines should be used.

`-simple` / `-composite`

This toggle indicates whether single object or object pair lines should be used.

`-matched` / `-unmatched`

This toggle indicates whether matched or unmatched object lines should be used.

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

`-model` value

This option specifies which model to use. **value** must be a string.

`-fcst_thr` value
`-obs_thr` value

These two options specify thresholds for forecast and observation objects to be used in the analysis, respectively.

`-fcst_var` value
`-obs_var` value

These options indicate the names of variables to be used in the analysis for forecast and observed fields.

`-fcst_lev` value
`-obs_lev` value

These options indicate vertical levels for forecast and observed fields to be used in the analysis.

<u>Multiple-set integer options</u>

The following options set various integer attributes. They can be set multiple times on the command line but must be separated by spaces. Each of the following options may only be indicated as an integer.

`-fcst_lead` value
`-lead2` value

The `–fcst_lead` option is an integer of the form HH[MMSS] specifying an (hour-minute-second) forecast lead time. The `–lead2` option specifies the time described in Table 6-2 (column 5 of MODE output files).

`-fcst_accum` value
`-obs_accum` value

This option is an integer of the form HHMMSS specifying an (hour-minute-second) forecast accumulation time.

`-fcst_rad` value
`-obs_rad` value

These options indicate the convolution radius used for forecast or observed objects, respectively.

<u>Integer max/min options</u>

These options set limits on various integer attributes. Leaving a maximum value unset means no upper limit is imposed on the value of the attribute. Similarly for minimum values.

`-area_min` value

`-area_max` value

These options are used to indicate minimum/maximum values for the area attribute to be used in the analysis.

`-area_filter_min` value
`-area_filter_max` value

These options are used to indicate minimum/maximum values accepted for the area filter. The area filter refers to the number of non-zero values of the raw data found within the object.

`-intersection_area_min` value
`-intersection_area_max` value

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

`-union_area_min` value
`-union_area_max` value

These options refer to the minimum/maximum union area values accepted for analysis.

`-symmetric_diff_min` value
`-symmetric_diff_max` value

These options refer to the minimum/maximum values for symmetric difference for objects to be used in the analysis.

Date/time max/min options
These options set limits on various date/time attributes. The values can be specified in one of three ways:

First, the options may be indicated by a string of the form YYYYMMDD_HHMMSS. This specifies a complete calendar date and time.

Second, they may be indicated by a string of the form YYYYMMDD_HH. Here, the minutes and seconds are assumed to be zero.

The third way of indicating date/time attributes is by a string of the form YYYYMMDD. Here, hours, minutes and seconds are assumed to be zero.

-fcst_valid_min yyyymmdd[_hh[mmss]]
-fcst_valid_max yyyymmdd[_hh[mmss]]

These options indicate minimum/maximum values for the forecast valid time.

-obs_valid_min yyyymmdd[_hh[mmss]]
-obs_valid_max yyyymmdd[_hh[mmss]]

These two options indicate minimum/maximum values for observation valid time.

Floating-point max/min options
Setting limits on various floating-point attributes. One may specify these as integers (i.e., without a decimal point), if desired. The following pairs of options indicate minimum and maximum values for each MODE attribute that can be described as a floating-point number. Please refer to Chapter 6 for a description of these attributes as needed.

-centroid_x_min value
-centroid_x_max value

-centroid_y_min value
-centroid_y_max value

-centroid_lat_min value
-centroid_lat_max value

-centroid_lon_min value
-centroid_lon_max value

-axis_ang_min value
-axis_ang_max value

-length_min value

-`length_max` value

-`width_min` value
-`width_max` value

-`curvature_min` value
-`curvature_max` value

-`curvature_x_min` value
-`curvature_x_max` value

-`curvature_y_min` value
-`curvature_y_max` value

-`complexity_min` value
-`complexity_max` value

-`intensity_10_min` value
-`intensity_10_max` value

-`intensity_25_min` value
-`intensity_25_max` value

-`intensity_50_min` value
-`intensity_50_max` value

-`intensity_75_min` value
-`intensity_75_max` value

-`intensity_90_min` value
-`intensity_90_max` value

-`intensity_user_min` value
-`intensity_user_max` value

`-intensity_sum_min` **value**
`-intensity_sum_max` **value**

`-centroid_dist_min` **value**
`-centroid_dist_max` **value**

`-boundary_dist_min` **value**
`-boundary_dist_max` **value**

`-convex_hull_dist_min` **value**
`-convex_hull_dist_max` **value**

`-angle_diff_min` **value**
`-angle_diff_max` **value**

`-area_ratio_min` **value**
`-area_ratio_max` **value**

`-intersection_over_area_min` **value**
`-intersection_over_area_max` **value**

`-complexity_ratio_min` **value**
`-complexity_ratio_max` **value**

`-percentile_intensity_ratio_min` **value**
`-percentile_intensity_ratio_max` **value**

`-interest_min` **value**
`-interest_max` **value**

Miscellaneous options

These options are used to indicate parameters that did not fall into any of the previous categories.

-mask_poly filename

This option indicates the name of a polygon mask file to be used for filtering. The format for these files is the same as that of the polyline files for the `point_stat`, `grid_stat`, or `mode` tools.

-help

This option prints the usage message.

8.3.2 *`mode_analysis` configuration file*

To use the MODE analysis tool, the user must un-comment the options in the configuration file to apply them and comment out unwanted options. The options in the configuration file for `mode_analysis` are the same as the MODE line options described in Section 8.3.1.

The parameters that are set in the configuration file either add to or override parameters that are set on the command line. For the "set string" and "set integer type" options enclosed in brackets, the values specified in the configuration file are added to any values set on the command line. For the toggle and min/max type options, the values specified in the configuration file override those set on the command line.

8.3.3 *MODE-Analysis tool output*

The output of the MODE Analysis tool is a self-describing tabular format written to standard output. The length and contents of the table vary depending on whether `–summary` or `–bycase` is selected. The contents also change for `–summary` depending on the fields selected by the user for output.

Chapter 9 – Scripting

9.1 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 9-1

```
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 9-1 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.

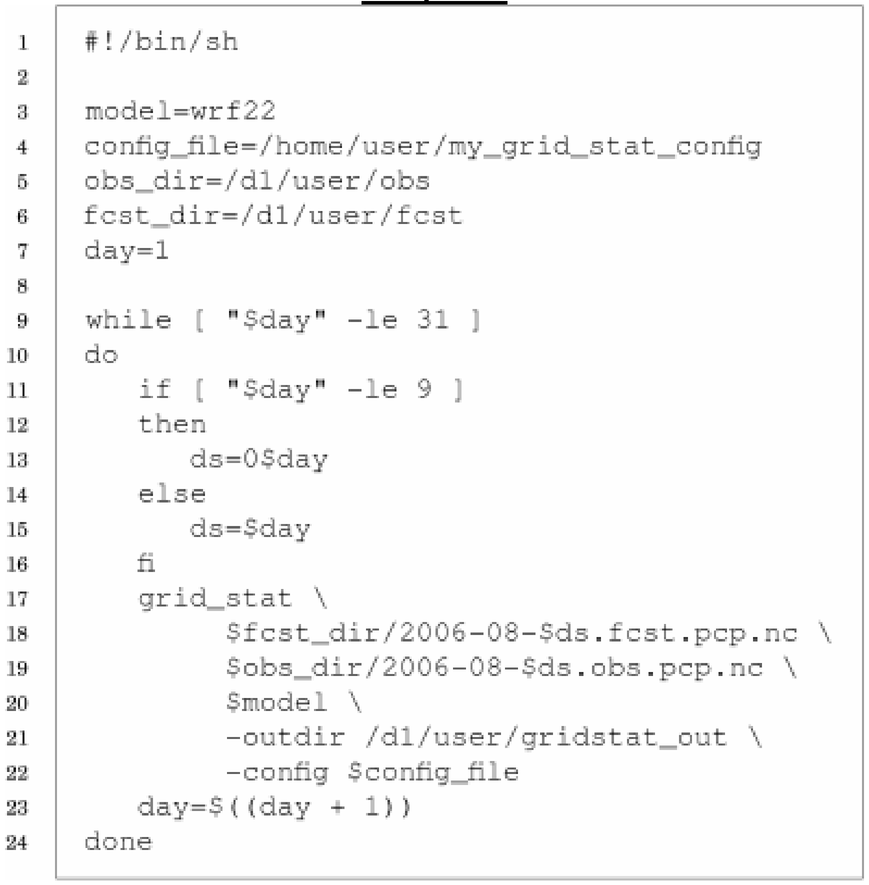
Line 7 starts the loop over the days in the month. The loop ends on line 23. Since `pcp_combine` needs dates and times formatted in a certain way, the variable `ds` (for daystring) is created, which contains the day of the month with a leading "0" if the day is less than 10. This happens in lines 9–14.

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 9-2



```
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 9-2, we define some useful variables: the model, the configuration file, and the directories for the observation and forecast files. As in Script 9-1, we loop through the days in the month. Line 7 initializes day to 1.

Lines 11–16 are copied from the Script 9-1. 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 9-1, the last line in the loop body (here, line 23) increments `day`.

9.2 Example scripts for use with MODE output files

In script 9-3 we use the c shell to run MODE with several different object definition parameters. We use nested `foreach` loops to go through (10 x 7 = 70) unique convolution radii and intensity threshold combinations. In the outer loop, we loop though ten convolution radii ranging from ‘00’ to ‘96’; in the inner loop, we loop through seven intensity thresholds. We assume that the observation grid and forecast grid are available in `pcp/st2/ST2ml_2005060100.g240.nc` and `pcp/wrf2caps/wrf2caps_2005053100.g240.f24.nc`, respectively.

Script 9-3

```
#!/bin/csh

# Loop through 10 convolution radii.
foreach conv_radius (00 04 08 12 16 24 32 48 64 96)

   # Set environmental variable conv_radius so it
   # may be used in the mode parameter file.
   setenv conv_radius $conv_radius

   # Set output directory.
   set dir = mode_output/${conv_radius}km
   # If output directory does not exist, create it.
   if (! -d $dir) mkdir $dir

   # Loop through 7 intensity thresholds.
   foreach conv_threshold (gt00. ge01. ge02. ge03. ge04. ge05. ge06.)

      # Set environmental variable conv_threshold so it
      # may be used in the mode parameter file.
      setenv conv_threshold \"$conv_threshold\"

      # Set output directory.
      set dir = mode_output/${conv_radius}km/${conv_threshold}
      # Create directory if it does not exist.
      if (! -d $dir) mkdir $dir

      # execute mode with output directory $dir
      mode pcp/wrf2caps/wrf2caps_2005053100.g240.f24.nc \
         pcp/st2/ST2ml_2005060100.g240.nc \
          mode_output/WrfModeConfig_default \
         -outdir $dir

   end
end
```

Line 1 is similar to Line 1 of scripts 9-1 and 9-2. It says that the c shell will interpret the script and the shell is found in /bin/csh. Comments are embedded in the script and are preceded by the number sign (#). Line 4 is the beginning of the outer loop, which runs through ten different convolution radii. These are presented in the form of 2-character strings, which are passed on to the environmental variable conv_radius (line 8). This environmental variable is passed on to the parameter file. An excerpt from the parameter file is shown below, in which we see how environmental variables are specified (lines 110-111). They are contained in curly brackets and preceded by a dollar sign. Line 11 of the shell script defines a path to the output directory. If the directory does not exist already, then it is created in line 13. The inner loop starts on line 16 and loops through seven intensity thresholds. Notice the use of backslashes before the double quote characters in line 20. This is necessary when the substitution will be for a string in the parameter file, such as for the parameters `fcst_conv_threshold` and `obs_conv_threshold` (lines 132-133 in the excerpt below). Again, the output directory is created if it doesn't exist (lines 23-25) and finally, mode is called in lines 28-31. The backslash characters at the end of the line are optional, but they are nice for breaking up a long command line into multiple lines. In practice, a return must immediately follow the backslash character. This script will execute mode 70 times with different object definition parameters for each execution. Each intensity threshold/ convolution radius combination will be output to its own directory.

If you copy and paste the script to your own text file, be sure you remove the line numbers. To execute it, change the permission of the text file to allow execution. Use the chmod command to do this (e.g., chmod u+x script9-3.csh).

<u>excerpt from mode parameter file to which script 9-3 refers</u>

```
107 // Radius (grid squares) for the circular convolution applied to the raw
108 // fcst and obs fields.
109 //
110 fcst_conv_radius     = ${conv_radius}/grid_res;
111 obs_conv_radius      = ${conv_radius}/grid_res;
112
113 //
114 // When performing the convolution step on points containing bad data,
115 // compute a ratio of the number of bad data points to the total number
116 // of points in the convolution area.  If that ratio is greater than this
117 // threshold, set the convolved field value to bad data.  Otherwise, use
118 // the computed convolution value.  Must be between 0 and 1.  Setting
119 // this threshold to 0 will have the effect of masking out bad data
120 // entirely from the object field.
121 //
122 bad_data_threshold   = 0.5;
123
124 //
125 // Apply a threshold to the convolved fcst and obs fields to define
126 // objects using the threshold values below.  The threshold values are
127 // specified as "xxT" where T is the threshold value and xx is one of:
128 //    'lt' for less than, 'le' for less than or equal to,
129 //    'eq' for equal to, 'ne' for not equal to,
130 //    'gt' for greater than, and 'ge' for greater than or equal to
```

```
//
fcst_conv_threshold  = ${conv_threshold};
obs_conv_threshold   = ${conv_threshold};
```

Chapter 10 – 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.

10.1 Grid-Stat tool examples

The plots in Figure 10-1 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.

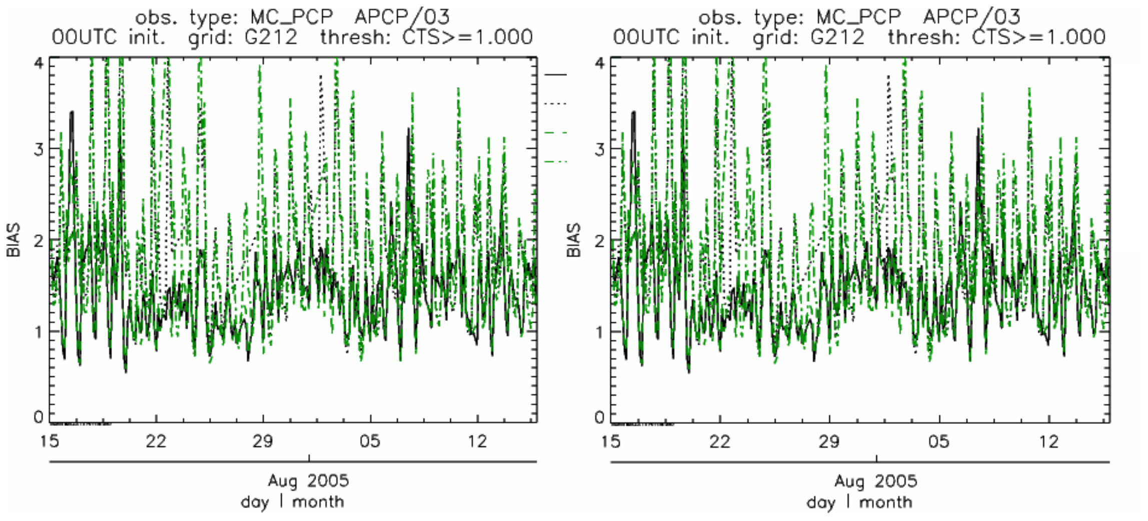


***Figure 10-1**: Time series of forecast bias and Gilbert Skill Score for several numerical models (differentiated by line-type and color) over a 32-day period in July/Aug 2005. Scores are based on the forecast of 1 mm or greater precipitation at 3-h intervals. Models were run for 24 h and were initiated every day at 00 UTC.*

A similar plot is shown in Fig. 10-2, 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.

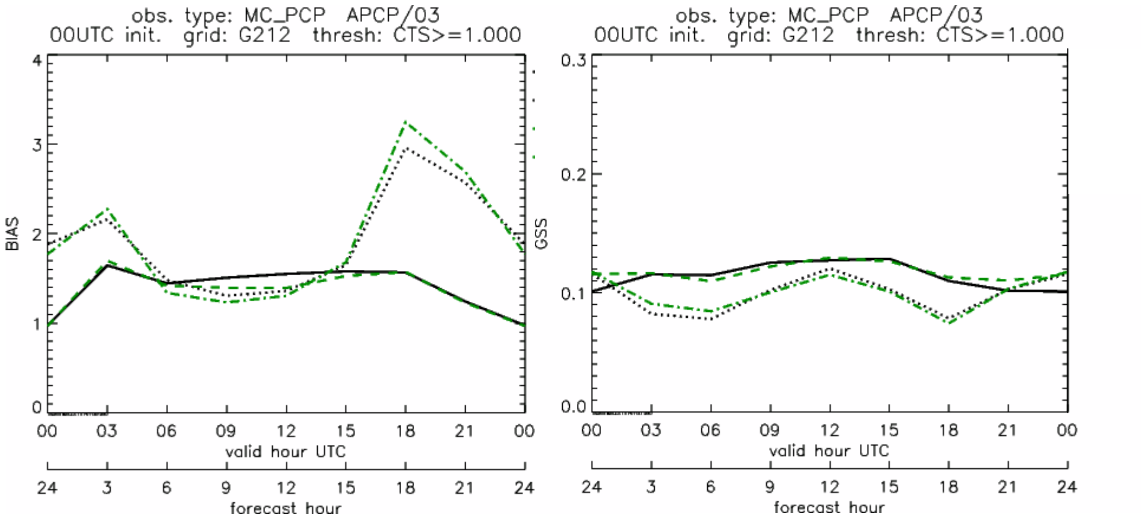


***Figure 10-2**: Time series of forecast area bias and Gilbert Skill Score for four model configurations (different lines) stratified by time-of-day. The data used to create these figures were the same as used for Fig. 10-1.*

10.2 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 10-3 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. 10-1 and 10-2. 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 Stage II radar-gauge precipitation. Note that for the 3-hr lead time, indicated as the 0300 UTC valid time in Fig. 10-3, 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.

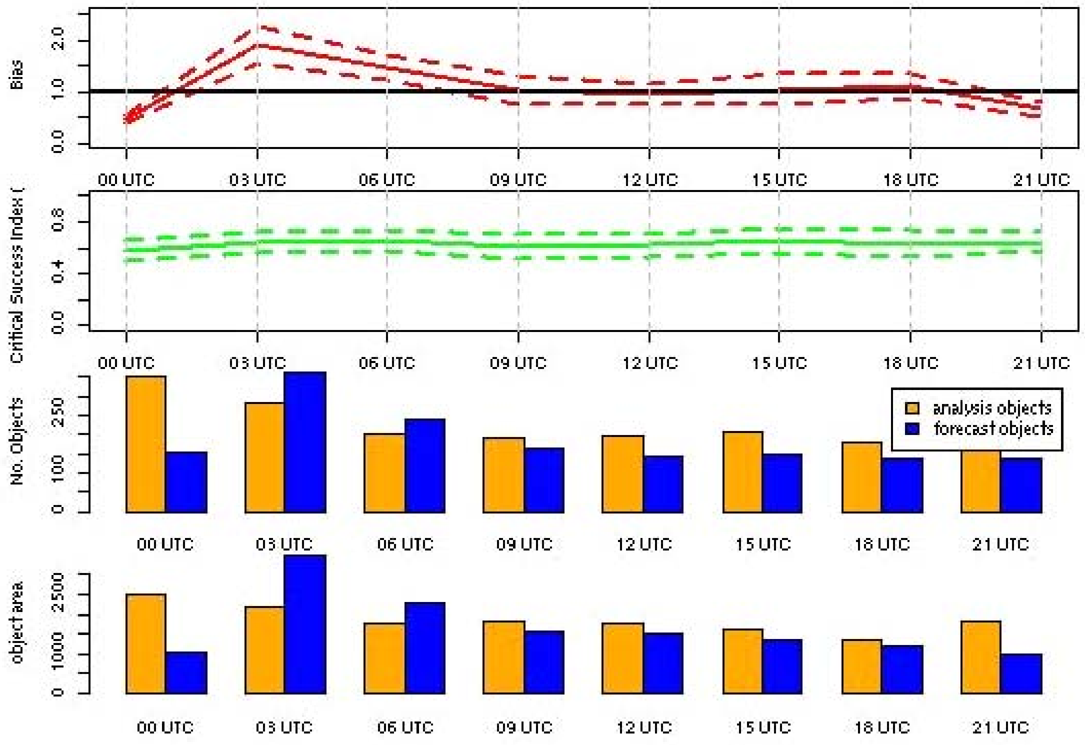
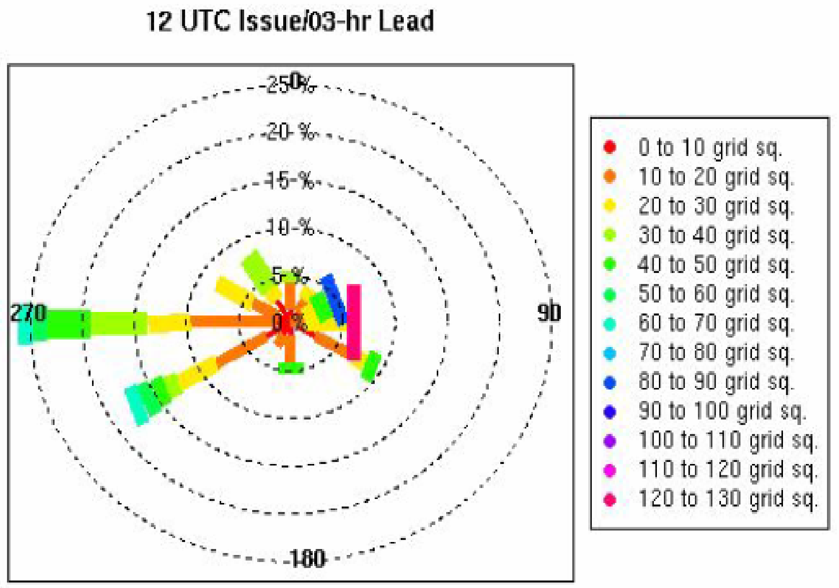


Figure 10-3: *Traditional verification scores applied to output of the MODE tool, computed by defining matched observed objects to be hits, unmatched observed objects to be misses, and unmatched forecast objects to be false alarms; weighted by object area. Bar plots show numbers (pennultimate row) and 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. 10-4, 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. 10-4 (top diagram) about 25% of the time the forecast object centroids are west of the observed object centroids, whereas in

Fig. 10-4 (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 10-4 (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.



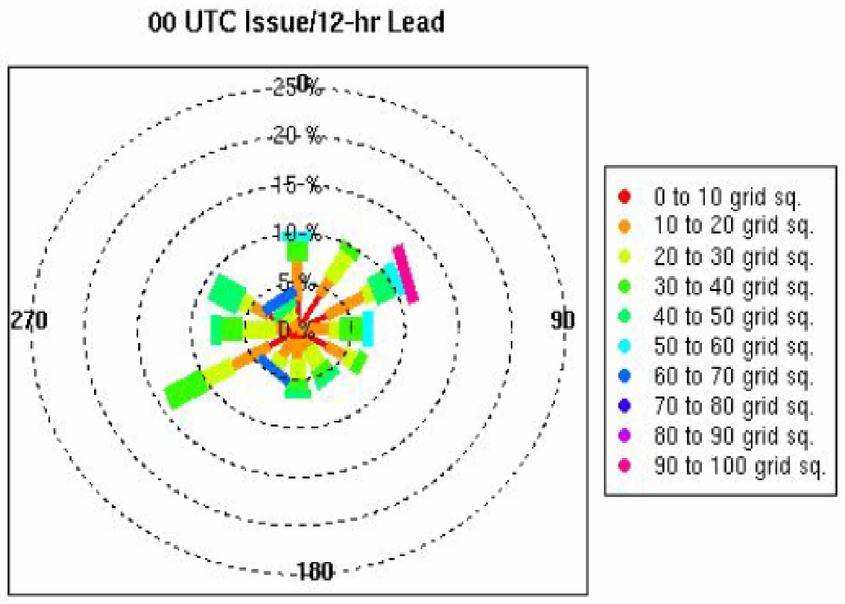


***Figure 10-4**: Circle histograms showing object centroid angles and distances (see text for explanation).*

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.

Ebert, E.E., Fuzzy verification of high-resolution gridded forecasts: a review and proposed framework. *Meteorological Applications,* **15**, 51-64.

Efron B. 2007: Correlation and large-scale significance testing. *Journal of the American Statistical Association,* **102**(477), 93-103.

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.

Roberts, N.M., and H.W. Lean, 2008: Scale-selective verification of rainfall accumulations from high-resolution forecasts of convective events. *Monthly Weather Review*, **136**, 78-97.

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. 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 future releases of MET.

Q. How does MET differ from the previously mentioned existing verification packages?
A. MET is an actively maintained, evolving software package that is being made freely available to the public through controlled version releases.

Q. How does the MODE tool differ from the Grid-Stat tool?
A. They offer different ways of viewing verification. The Grid-Stat tool provides traditional verification statistics, while MODE provides specialized spatial statistics.

Q. Will the MET work on data in native model coordinates?
A. No – it will not. In the future, we may add options to allow additional model grid coordinate systems.

Q. How do I get help if my questions are not answered in the User's Guide?
A. First, look on our website http://www.dtcenter.org/met/users. If that doesn't answer your question, then ***email:*** ***met_help@ucar.edu.***

Q. Where are the graphics?
A. Currently, very few graphics are included. Further graphics support will be made available in the future on the MET website.

A.2 Troubleshooting

The first place to look for help with individual commands is this user's guide or the usage statements that are provided with the tools. Usage statements for the individual commands (`grid_stat`, `point_stat`, `mode`, `pcp_combine`, `pb2nc`, `ascii2nc`, `vsdb_analysis`, and `mode_analysis`) are available by simply typing the name of the executable in MET's `bin/` directory. Example scripts available in the MET's `scripts/` directory show examples of how one might use these commands on example datasets. Here are suggestions on other things to check if you are having problems installing or running MET.

MET won't compile

- Are you using the correct version of the Makefile (Makefile_gnu for using the GNU compilers; Makefile_pgi for the PGI compilers, and Makefile_ibm for running on an IBM)
- In your configured Makefile, did you accidently insert any blank characters at the end of a line. Doing so may cause the compiler options to be passed incorrectly.
- 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 using the same set of compilers used to build MET?
- Do you have the correct F2C (FORTRAN to C) header? This may be either "libf2c.a" or "libg2c.a" depending on your system.
- Are you using NetCDF version 3.4 or version 4? Currently, only NetCDF version 3.6 can be used with MET.

Grid_stat won't run

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

MODE won't run

- 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

- Have you run `pb2nc` first on your PrepBufr observation data?

General troubleshooting

- For configuration files used, make certain to use empty square brackets (e.g. [ ]) to indicate no stratification is desired. Do NOT use empty double quotation marks inside square brackets (e.g. [""]).
- Have you designated all the required command line arguments?

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

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

The NCEP grids that 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 – Verification Measures

This appendix provides specific information about the many verification statistics and measures that are computed by MET. These measures are categorized into measures for categorical (dichotomous) variables; measures for continuous variables; and measures for neighborhood methods. While the continuous and categorical statistics are computed by both the Point-Stat and Grid-Stat tools, the neighborhood verification measures are only provided by the Grid-Stat tool.

C.1 MET verification measures for categorical (dichotomous) variables

The verification statistics for dichotomous variables are formulated using a contingency table such as the one shown in Table C-1. In this table f represents the forecasts and o represents the observations; the two possible forecast and observation values are represented by the values 0 and 1. The values in Table C-1 are counts of the number of occurrences of the four possible combinations of forecasts and observations.

Table C-1*: 2x2 contingency table in terms of counts. The 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 C-1 can also be used to compute the F, O, and H relative frequencies that are produced by the NCEP Verification System, and the Point-Stat tool provides an option to produce the statistics in this form. In terms of the other statistics computed by the Point-Stat tool, F is equivalent to the Mean Forecast; H is equivalent to POD; and O is equivalent to the Base Rate. All of these statistics are defined in the subsections below. The Point-Stat tool also provides the total number of observations, T.

The categorical verification measures produced by the Point-Stat and Grid-Stat tools are described in the following subsections. They are presented in the order shown in Tables 4-3 through 4-8.

TOTAL

The total number of forecast-observation pairs, T.

Base rate

Called "O_RATE" in FHO output (Table 4-3)
Called "OY_TP" in CTP output (Table 4-5)
Called "BASER" in CTS output (Table 4-8)

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

Called "F_RATE" in FHO output (Table 4-3);
Called "FY_TP" in CTP output (Table 4-5);
Called "FMEAN" in CTS output (Table 4-8)

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

Called "ACC" in CTS output (Table 4-8)

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
Called “FBIAS” in CTS output (Table 4-8)

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 $\mathrm{Bias} = \frac{n_{11} + n_{10}}{n_{11} + n_{01}} = \frac{n_{1.}}{n_{.1}}$. 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)
Called “FY_OY_OP” in COP output (Table 4-7)
Called “PODY” in CTS output (Table 4-8)

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)
Called “FY_ON_OP” in COP output (Table 4-7)
Called “POFD” in CTS output (Table 4-8)

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**[2]. POFD ranges from 0 to 1; a perfect forecast would have POFD = 0.

Probability of Detection of the non-event (PODn)
Called “FN_ON_OP” in COP output (Table 4-7)
Called “PODN” in CTS output (Table 4-8)

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.

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

False Alarm Ratio (FAR)
Called "FY_ON_FP" in CFP output (Table 4-6)
Called "FAR" in CTS output (Table 4-8)

FAR is defined as $\text{FAR} = \frac{n_{10}}{n_{11} + n_{10}} = \frac{n_{10}}{n_{1.}}$. 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)
Called "CSI" in CTS output (Table 4-8)

CSI is defined as $\text{CSI} = \frac{n_{11}}{n_{11} + n_{10} + n_{01}}$. It is the ratio of the number of times the event was correctly forecasted to occur to the number of times it was either forecasted or occurred. CSI ignores the "correct rejections" category (i.e., n_{00}). CSI is also known as the **Threat Score (TS)**. CSI can also be written as a nonlinear combination of POD and FAR, and is strongly related to Frequency Bias and the Base Rate.

Gilbert Skill Score (GSS)
Called "GSS" in CTS output (Table 4-8)

GSS is based on the CSI, corrected for the number of hits that would be expected by chance. In particular, $\text{GSS} = \frac{n_{11} - C_1}{n_{11} + n_{10} + n_{01} - C_1}$, where $C_1 = \frac{(n_{11} + n_{10})(n_{11} + n_{01})}{T} = \frac{n_{1.}n_{.1}}{T}$.
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)
Called "HK" in CTS output (Table 4-8)

H-K is defined as $\text{H-K} = \frac{n_{11}n_{00} - n_{10}n_{01}}{(n_{11} + n_{01})(n_{10} + n_{00})}$. More simply, $\text{H-K} = \text{POD} - \text{POFD}$. H-K is also known as the **True Skill Statistic (TSS)** and less commonly (although perhaps more properly) as the **Peirce Skill Score**. H-K measures the ability of the forecast to discriminate between (or correctly classify) events and non-events. H-K values range between -1 and 1. A value of 0 indicates no skill; a perfect forecast would have H-K = 1.

Heidke Skill Score (HSS)
Called "HSS" in CTS output (Table 4-8)

HSS is a skill score based on Accuracy, where the Accuracy is corrected by the number of correct forecasts that would be expected by chance. In particular,

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

<u>Odds Ratio (OR)</u>
Called "ODDS" in CTS output (Table 4-8)

OR measures the ratio of the odds of a forecast of the event being correct to the odds of a forecast of the event being wrong. OR is defined as $\text{OR} = \frac{n_{11} \times n_{00}}{n_{01} \times n_{10}} = \frac{\left(\frac{\text{POD}}{1 - \text{POD}}\right)}{\left(\frac{\text{POFD}}{1 - \text{POFD}}\right)}$.

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).

C.2 MET verification measures for continuous variables

For continuous variables, many verification measures are based on the forecast error (i.e., $f - o$). However, it also is of interest to investigate characteristics of the forecasts, and the observations, as well as their relationship. These concepts are consistent with the general framework for verification outlined by Murphy and Winkler (1987). The statistics produced by MET for continuous forecasts represent this philosophy of verification, which focuses on a variety of aspects of performance rather than a single measure.

The verification measures currently evaluated by the Point-Stat tool are defined and described in the subsections below. In these definitions, f represents the forecasts, o represents the observation, and n is the number of forecast-observation pairs.

<u>Mean forecast</u>
Called "FBAR" in CNT output (Table 4-9)
Called "FBAR" in SL1L2 output (Table 4-10)

The sample mean forecast, FBAR, is defined as $\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f_i$

<u>Mean observation</u>
Called "OBAR" in CNT output (Table 4-9)
Called "FBAR" in SL1L2 output (Table 4-10)

The sample mean observation is defined as $\bar{o} = \frac{1}{n}\sum_{i=1}^{n} o_i$, respectively.

<u>Forecast standard deviation</u>
Called "FSTDEV" in CNT output (Table 4-9)

The sample variance of the forecasts is defined as $s_f^2 = \frac{1}{n-1}\sum_{i=1}^{T}(f_i - \bar{f})^2$.

The forecast standard deviation is defined as $s_f = \sqrt{s_f^2}$.

<u>Observation standard deviation</u>
Called "OSTDEV" in CNT output (Table 4-9)

The sample variance of the observations is defined as $s_o^2 = \frac{1}{n-1}\sum_{i=1}^{n}(o_i - \bar{o})^2$.

The observed standard deviation is defined as $s_o = \sqrt{s_o^2}$.

<u>Pearson Correlation Coefficient</u>
Called "PR_CORR" in CNT output (Table 4-9)

The **Pearson correlation coefficient**, r, measures the strength of linear association between the forecasts and observations. The Pearson correlation coefficient is defined as

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

r can range between -1 and 1; a value of 1 indicates perfect correlation and a value of -1 indicates perfect negative correlation. A value of 0 indicates that the forecasts and observations are not correlated.

Spearman rank correlation coefficient (ρ_s)
Called "SP__CORR" in CNT (Table 4-9)

The **Spearman rank correlation coefficient (ρ_s)** is a robust measure of association that is based on the ranks of the forecast and observed values rather than the actual values. That is, the forecast and observed samples are ordered from smallest to largest and rank values (from 1 to n, where n is the total number of pairs) are assigned. The pairs of forecast-observed ranks are then used to compute a correlation coefficient, as shown for the Pearson correlation coefficient, r.

A simpler formulation of the Spearman-rank correlation is based on the differences between the each of the pairs of ranks (denoted as d_i):

$$\rho_s = \frac{6}{n(n^2-1)}\sum_{i=1}^{n} d_i^2$$

Like r, the Spearman rank correlation coefficient ranges between -1 and 1; a value of 1 indicates perfect correlation and a value of -1 indicates perfect negative correlation. A value of 0 indicates that the forecasts and observations are not correlated.

Kendall's tau statistic (τ)
Called "KT_CORR" in CNT output (Table 4-9)

Kendall's tau statistic (τ) is a robust measure of the level of association between the forecast and observation pairs. It is defined as

$$\tau = \frac{N_C - N_D}{n(n-1)/2}$$

where N_C is the number of "concordant" pairs and N_D is the number of "discordant" pairs. Concordant pairs are identified by comparing each pair with all other pairs in the sample; this can be done most easily by ordering all of the (f_i, o_i) pairs according to f_i, in which case the o_i values won't necessarily be in order. The number of concordant matches of a particular pair with other pairs is computed by counting the number of pairs (with larger f_i values) for which the value of o_i for the current pair is exceeded (that is, pairs for which the values of f and o are both larger than the value for the current pair). Once this is done, N_c is computed by summing the counts for all pairs. The total number of possible pairs is $n(n-1)/2$; thus, the number of discordant pairs is $N_D = n(n-1)/2 - N_C$.

Like r and ρ_s, Kendall's tau (τ) ranges between -1 and 1; a value of 1 indicates perfect association (concordance) and a value of -1 indicates perfect negative association. A value of 0 indicates that the forecasts and observations are not associated.

Mean Error (ME)
Called "ME" in CNT output (Table 4-9)

The Mean Error, **ME**, is a measure of overall bias for continuous variables; in particular ME = Bias. It is defined as

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

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

Multiplicative Bias
Called "MBIAS" in CNT output (Table 4-9)

Multiplicative bias is simply the ratio of the means of the forecasts and the observations:

$$\mathrm{MBIAS} = \frac{\bar{f}}{\bar{o}}$$

Mean-squared error (MSE)
Called "MSE" in CNT output (Table 4-9)

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

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

Root-mean-squared error (RMSE)
Called "RMSE" in CNT output (Table 4-9)

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

Standard deviation of the error
Called "ESTDEV" in CNT output (Table 4-9)

Bias-Corrected MSE
Called "BCMSE" in CNT output (Table 4-9)

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

$\mathrm{MSE} = (\overline{f} - \overline{o})^2 + s_f^2 + s_o^2 - 2 s_f s_o r_{fo}$, where $\overline{f} - \overline{o} = \mathrm{ME}$ and $s_f^2 + s_o^2 - 2 s_f s_o r_{fo}$ is the estimated variance of the error, s_{f-o}^2. Thus, $\mathrm{MSE} = \mathrm{ME}^2 + s_{f-o}^2$. To understand the behavior of **MSE**, it is important to examine *both* of the terms of **MSE**, rather than examining **MSE** alone. Moreover, **MSE** can be strongly influenced by **ME**, as shown by this decomposition.

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

Note that the standard deviation of the error (ESTDEV) is sometimes called the “Bias-corrected MSE” (BCMSE) because it removes the effect of overall bias from the forecast-observation squared differences.

Mean Absolute Error (MAE)
Called “MAE” in CNT output (Table 4-9)

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

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

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

Percentiles of the errors
Called “E10”, “E25”, “E50”, “E75”, “E90” in CNT output (Table 4-9)

Percentiles of the errors provide more information about the distribution of errors than can be obtained from the mean and standard deviations of the errors. Percentiles are computed by ordering the errors from smallest to largest and computing the rank location of each percentile in the ordering, and matching the rank to the actual value. Percentiles can also be used to create box plots of the errors. In MET, the 0.10th, 0.25th, 0.50th, 0.75th, and 0.90th quantile values of the errors are computed.

Scalar L1 and L2 values
Called “FBAR”, “OBAR”, “FOBAR”, “FFBAR”, and “OOBAR” in SL1SL2 output (Table 4-10)

These statistics are simply the 1st and 2nd moments of the forecasts, observations and errors:

$$\text{FBAR} = \text{Mean}(f) = \bar{f} = \frac{1}{n}\sum_{i=1}^{n} f_i$$

$$\text{OBAR} = \text{Mean}(o) = \frac{1}{n}\sum_{i=1}^{n} o_i$$

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

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

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

Some of the other statistics for continuous forecasts (e.g., RMSE) can be derived from these moments.

<u>Scalar anomaly L1L2 values</u>
Called FABAR, OABAR, FOABAR, FFABAR, OOABAR in SAL1L2 output (Table 4-11)

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

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

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

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

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

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

<u>Vector L1 and L2 values</u>
Caleld “UFBAR”, “VFBAR”, “UOBAR”, “VOBAR”, “UVFOBAR”, “UVFFBAR”, “UVOOBAR” in VL1L2 output (Table 4-12)

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:

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

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

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

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

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

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

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

Vector anomaly L1 and L2 values

Called “UFABAR”, “VFABAR”, “UOABAR”, “VOABAR”, “UVFOABAR”, “UVFFABAR”, “UVOOABAR” in VAL1L2 output (Table 4-13)

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

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

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

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

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

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

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

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

C.3 MET verification measures for neighborhood methods

The results of the neighborhood verification approaches that are included in the Grid-Stat tool are summarized using a variety of measures. These measures include the Fractions Skill Score (FSS) and the Fractions Brier Score (FBS). MET also computes traditional contingency table statistics for each combination of threshold and neighborhood window size.

The traditional contingency table statistics computed by the Grid-Stat neighborhood tool, and included in the NBRCTS output, are listed below:

- Base Rate (called "BASER" in Table 5-3)
- Mean Forecast (called "FMEAN" in Table 5-3)
- Accuracy (called "ACC" in Table 5-3)
- Frequency Bias (called "FBIAS" in Table 5-3)
- Probability of Detection (called "PODY" in Table 5-3)
- Probability of Detection of the non-event (called "PODN" in Table 5-3)
- Probability of False Detection (called "POFD" in Table 5-3)
- False Alarm Ratio (called "FAR" in Table 5-3)
- Critical Success Index (called "CSI" in Table 5-3)
- Gilbert Skill Score (called "GSS" in Table 5-3)
- Hanssen-Kuipers Discriminant (called "H-K" in Table 5-3)
- Heidke Skill Score (called "HSS" in Table 5-3)
- Odds Ratio (called "ODDS" in Table 5-3)

All of these measures are defined in Section C1 of Appendix C.

In addition to these standard statistics, the neighborhood analysis provides two additional continuous measures, the ***Fractions Brier Score*** and the ***Fractions Skill Score***. These measures are defined here, but are explained in much greater detail in Ebert (2008) and Roberts and Lean (2008). Roberts and Lean (2008) also present an application of the methodology.

Fractions Brier Score

Called "FBS" in NBRCNT output (Table 5-4)

The Fractions Brier Score (FBS) is defined as $\mathrm{FBS} = \frac{1}{N}\sum_{N}(\langle P_f \rangle_s - \langle P_o \rangle_s)^2$, where N is the number of neighborhoods; $\langle P_f \rangle_s$ is the proportion of grid boxes within a forecast neighborhood where the prescribed threshold was exceeded (i.e., the proportion of grid boxes that have forecast events); and $\langle P_o \rangle_s$ is the proportion of grid boxes within an observed neighborhood where the prescribed threshold was exceeded (i.e., the proportion of grid boxes that have observed events).

Fractions Skill Score

Called "FSS" in NBRCNT output (Table 5-4)

The Fractions Skill Score (FSS) is defined as $\mathrm{FSS} = \frac{\mathrm{FBS}}{\frac{1}{N}\left[\sum_N \langle P_f \rangle_s^2 - \sum_N \langle P_o \rangle_s^2\right]}$, where the denominator represents the worst possible forecast (i.e., with no overlap between forecast and observed events). FSS ranges between 0 and 1, with 0 representing no overlap and 1 representing complete overlap between forecast and observed events, respectively.

Appendix D – Confidence Intervals

A single verification statistic is statistically meaningless without associated uncertainty information in accompaniment. There can be numerous sources of uncertainty associated with such a statistic including observational, physical uncertainties about the underlying processes governing the equation, sample uncertainty, etc. Although all of the sources of uncertainty can be important, the most heavily researched, and easiest to calculate, is that of sampling uncertainty. It is this source of uncertainty that can presently be obtained with MET, and the techniques for deriving these estimates are described here. Sampling uncertainty through MET is gleaned by way of confidence intervals (CIs) as these are generally most informative. A $(1-\alpha)\cdot 100\%$ confidence interval is interpreted, somewhat awkwardly, in the following way. If the test were repeated 100 times (so that we have 100 such intervals), then we expect the true value of the statistics to fall inside $(1-\alpha)\cdot 100$ of these intervals. For example, if $\alpha = 0.05$ then we expect the true value to fall within 95 of the intervals.

There are two main types of CIs available with MET: parametric and non-parametric. All of the parametric intervals used with MET rely on the underlying sample (or the errors, $F-O$) to be at least approximately independent and normally distributed. Future releases will allow for some types of dependency in the sample. The non-parametric techniques utilize what is known in the statistical literature as bootstrap resampling, which does not rely on any distributional assumptions for the sample; the assumption is that the sample is representative of the population. Bootstrap CIs can be inaccurate if the sample is not independent, but there are ways of accounting for dependence with the bootstrap, some of which will be added to MET in future releases. Details about which verification statistics have parametric CIs in MET are described next, and it should be noted that the bootstrap can be used for any statistic, though care should be taken in how it is carried out, and this is described subsequently.

The most commonly used confidence interval about an estimate for a statistic (or parameter), θ, is given by the normal approximation

$$\hat{\theta} \pm z_{\alpha/2} \cdot V(\theta), \tag{D.1}$$

where z_{α} is the α-th quantile of the standard normal distribution, and $V(\theta)$ is the standard error of the statistic (or parameter), θ. For example, the most common example is for the mean of a sample, $X_1,\ldots,X_n$, of independent and identically distributed (iid) normal random variables with mean μ and variance σ^2. Here, the mean is estimated by $\frac{1}{n}\sum_{i=1}^{n} X_i = \bar{X}$, and the standard error is just the standard deviation of the random variables divided by the square root of the sample size. That is,

$V(\theta) = V(\bar{X}) = \frac{\sigma}{\sqrt{n}}$, and this must be estimated by $\hat{V}(\bar{X})$, which is obtained here by replacing σ by its estimate, $\hat{\sigma}$, where $\hat{\sigma}^2 = \sum_{i=1}^{n} (X_i - \bar{X})^2 / (n-1)$.

Mostly, the normal approximation is used as an asymptotic approximation. That is, the interval (D.1) may only be appropriate for large n. For small n, the mean has an interval based on the Student's t distribution with $n-1$ degrees of freedom. Essentially, $z_{\alpha/2}$ of (D.1) is replaced with the $\alpha/2$ quantile of this t distribution. That is, the interval is given by

$$\hat{\mu} \pm t_{\alpha/2,n-1} \cdot \frac{\sigma}{\sqrt{n}}, \qquad \text{(D.2)}$$

where again, σ is replaced by its estimate, $\hat{\sigma}$, as described above.

Table 1 summarizes the verification statistics in MET that have normal approximation CIs given by (D.1) along with their corresponding standard error estimates, $\hat{V}(\theta)$. It should be noted that for the first two rows of this table (i.e., Forecast/Observation Mean and Mean error) MET also calculates the interval (D.2) for small sample sizes.

Table D1: Verification statistics with normal approximation CIs provided given by (D.1) provided in MET along with their associated standard error estimate.

$\hat{\theta}$	$\hat{V}(\theta)$
Forecast/Observation Mean	$\hat{V}(\bar{X}) = \frac{\hat{\sigma}_X}{\sqrt{n}}$, where $\hat{\sigma}_X$ emphasizes that this is the estimated standard deviation of the underlying sample.
Mean error	$\hat{V}(\bar{F} - \bar{O}) = \frac{\hat{\sigma}_{F-O}}{\sqrt{n}}$, where $\hat{\sigma}_{F-O}$ emphasizes that this is the estimated standard deviation of the errors, $F - O$.
Peirce Skill Score (PSS)	$\hat{V}(PSS) = \sqrt{\frac{H(1-H)}{n_H} + \frac{F(1-F)}{n_F}}$, where H is the hit rate, F the false alarm rate, n_H the number of hits and misses, and n_F the number of false alarms and correct negatives.
Logarithm of the odds ratio (OR)	$\hat{V}(\ln(OR)) = \sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}$, where the values in the denominators are the usual contingency table counts.

Other statistics in MET having parametric CIs that rely on the underlying sample to be at least approximately iid normal, but have a different form derived from the normality assumtion on the sample include the variance, standard deviation, and the linear correlation coefficient. These are addressed subsequently.

Generally, the normal interval (D.1) is appropriate for statistics of continuous variables, but a limit law for the binomial distribution allows for use of this interval with proportions. The most intuitive estimate for $V(\theta)$ in this case is given by $\hat{V}(p)\sqrt{\hat{p}(1-\hat{p})/n}$. However, this only applies when the sample size is large. A better approximation to the CI for proportions is given by Wilson's interval, which is

$$\frac{\hat{p}+z_{\alpha/2}^2+z_{\alpha/2}\sqrt{\hat{p}(1-\hat{p})/(4n)}}{1+z_{\alpha/2}^2/n}, \tag{D.3}$$

where $\hat{p}$ is the estimated proportion (e.g., hit rate, false alarm rate, PODy, PODn, etc.). Because this interval (D.3) generally works better than the more intuitive normal approximation interval for both large and small sample sizes, this is the interval employed by MET.

The forecast/observation variance has CIs derived from the underlying sample being approximately iid normal with mean μ and variance σ^2. The lower and upper limits for the interval are given by

$$\ell(\sigma^2)=\frac{(n-1)s^2}{\chi^2_{\alpha/2,n-1}} \text{ and}$$

$$u(\hat{\sigma}^2)=\frac{(n-1)s^2}{\chi^2_{1-\alpha/2,n-1}}, \tag{D.4}$$

respectively, where $\chi^2_{\alpha,\nu}$ is the α-th quantile of the chi-square distribution with v degrees of freedom. Taking the square roots of the limits in (D.4) yields the CI for the forecast/observation standard deviation.

Finally, the linear correlation coefficient has limits given by

$$\left(\frac{e^{2c_\ell}-1}{e^{2c_\ell}+1},\frac{e^{2c_u}-1}{e^{2c_u}+1}\right), \tag{D.5}$$

where $c_\ell=\nu-\frac{z_{\alpha/2}}{\sqrt{n-3}}$ and $c_u=\nu+\frac{z_{\alpha/2}}{\sqrt{n-3}}$.

All other verification scores with CIs in MET must be obtained through bootstrap resampling. However, it is also possible to obtain bootstrap CIs for any of the statistics given above, and indeed it has been proven that the bootstrap intervals have better accuracy for the mean than the normal approximation. The bootstrap algorithm is described below.

1. Assume the sample is representative of the population.
2. Resample *with replacement* from the sample (see text below).
3. Estimate the parameter(s) of interest for the current replicated sample.
4. Repeat steps 2 and 3 numerous times, say B times, so that you now have a sample of size B of the parameter(s).
5. Calculate CIs for the parameters directly from the sample (see text below for more details)

Typically, a simple random sample is taken for step 2, and that is how it is done in MET. As an example of what happens in this step, suppose our sample is X_1, X_2, X_3, X_4. Then, one possible replicate might be X_2, X_2, X_2, X_4. Usually one samples *m* = *n* points in this step, but there are cases where one should use *m* < *n*. For example, when the underlying distribution is heavy-tailed, one should use a smaller size m than n (e.g., the closest integer value to the square root of the original sample size).

There are numerous ways to construct CIs from the sample obtained in step 4. MET allows for two of these procedures: the percentile and the BCa. The percentile is the most commonly known method, nd the simplest to understand. It is merely the $\alpha/2$ and $1-\alpha/2$ percentiles from the sample of statistics. Unfortunately, however, it has been shown that this interval is too optimistic in practice (i.e., it doesn't have accurate coverage). One solution is to use the BCa method, which is very accurate, but it is also computationally intensive. This method adjusts for bias and non-constant variance, and yields the percentile interval in the event that the sample is unbiased with constant variance.

If there is dependency in the sample, then it is prudent to account for this dependency in some way. One method that does not make a lot of assumptions is circular block bootstrapping. This is not currently implemented in MET, but will be available in a future release. At that time, the method will be explained more fully here, but until then consult Gilleland (2008, unpublished) for more details.