Model Evaluation Tools Version 4.0 (METv4.0)

User's Guide 4.0

Developmental Testbed Center

Boulder, Colorado, USA

June 2012

Contents

Section **Page**

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. Previous releases of MET have occurred each year since 2008. This documentation describes the 4.0 release from 2012 that includes refined and streamlined computations as well as corrections to some 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, Randy Bullock, Paul Oldenburg, Anne Holmes, Tara Jensen, Lacey Holland, Barbara Brown, Tressa Fowler, David Ahijevych, Eric Gilleland and Bonny Strong.

New for MET v4.0

Users will notice two major upgrades for METv4.0, support for GRIB2 and a restructuring of the MET configuration files. A list these and other upgrades is included below.

Major Upgrades:

1. Added optional support for GRIB2. This support depends on the external NCEP GRIB2 C-library. Support for GRIB2 is enabled by setting the "WITH_GRIB2" flag in "user_defs.mk".
2. Restructured the MET configuration files to make them more readable and give the user finer control over the verification tasks to be performed.
3. See "data/config/README" for more detail on the changes.

Enhancements to Existing Tools:

1. Distributed latest set of bug fixes for METv3.1.
2. Enhanced Point-Stat and Ensemble-Stat to apply filtering logic when encountering duplicate point observations at the same location. See the "duplicate_flag" section in "data/config/README" for more detail.
3. Enhanced PCP-Combine to run on all supported gridded data file types, rather than just GRIB1. Added -config command line option to specify the field to be processed.
4. Enhanced STAT-Analysis and MODE-Analysis by adding support for the

5. -fcst_valid_hour and -obs_valid_hour filtering options.
6. Enhanced STAT-Analysis to aggregate multiple NBRCNT lines together for the "aggregate" job type.
7. Enhanced handling of GRIB1 and GRIB2 table information by reading it from data files at runtime rather than having it hard-coded in header files.
8. For Point-Stat, Grid-Stat, Wavelet-Stat, Ensemble-Stat, and MODE, moved
9. Support for the -fcst_valid, -fcst_lead, -obs_valid, and -obs_lead filtering options from the command line to the configuration files.
10. Updated the WWMCA tools to ensure that pixel age masking is optional.
11. Fixed STAT-Analysis bug in the computation of the GO Index.
12. Fixed Ensemble-Stat bug in the handling of missing data.
13. Fixed library code bug when applying polyline masks to global datasets.

Enhancement to the Build Process:

1. Defined unit tests for each of the MET tools and major library functionality. These tests are not distributed with the code but are routinely run during development to ensure that new development does not break existing functionality.

TERMS OF USE

IMPORTANT!

USE OF THIS SOFTWARE IS SUBJECT TO THE FOLLOWING TERMS AND CONDITIONS:

1. **License.** Subject to these terms and conditions, University Corporation for Atmospheric Research (UCAR) grants you a non-exclusive, royalty-free license to use, create derivative works, publish, distribute, disseminate, transfer, modify, revise and copy the Model Evaluation Tools (MET) software, in both object and source code (the "Software").

   You shall not sell, license or transfer for a fee the Software, or any work that in any manner contains the Software.

2. **Disclaimer of Warranty on Software.** Use of the Software is at your sole risk. The Software is provided "AS IS" and without warranty of any kind and UCAR EXPRESSLY DISCLAIMS ALL WARRANTIES AND/OR CONDITIONS OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT OF A THIRD PARTY'S INTELLECTUAL PROPERTY, MERCHANTABILITY OR SATISFACTORY QUALITY AND FITNESS FOR A PARTICULAR PURPOSE. THE PARTIES EXPRESSLY DISCLAIM THAT THE UNIFORM COMPUTER INFORMATION TRANSACTIONS ACT (UCITA) APPLIES TO OR GOVERNS THIS AGREEMENT. No oral or written information or advice given by UCAR or a UCAR authorized representative shall create a warranty or in any way increase the scope of this warranty. Should the Software prove defective, you (and neither UCAR nor any UCAR representative) assume the cost of all necessary correction.

3. **Limitation of Liability.** UNDER NO CIRCUMSTANCES, INCLUDING NEGLIGENCE, SHALL UCAR BE LIABLE FOR ANY DIRECT, INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES INCLUDING LOST REVENUE, PROFIT OR DATA, WHETHER IN AN ACTION IN CONTRACT OR TORT ARISING OUT OF OR RELATING TO THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF UCAR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

4. **Compliance with Law.** All Software and any technical data delivered under this Agreement are subject to U.S. export control laws and may be subject to export or import regulations in other countries. You agree to comply strictly with all applicable laws and regulations in connection with use and distribution of the Software, including export control laws, and you acknowledge that you have responsibility to obtain any required license to export, re-export, or import as may be required.

5. **No Endorsement/No Support.** The names UCAR/NCAR, National Center for Atmospheric Research and the University Corporation for Atmospheric Research may not be used in any advertising or publicity to endorse or promote any products or commercial entity unless specific written permission is obtained from UCAR. The Software is provided without any support or maintenance, and without any obligation to provide you with modifications, improvements, enhancements, or updates of the Software.

6. **Controlling Law and Severability.** This Agreement shall be governed by the laws of the United States and the State of Colorado. If for any reason a court of competent jurisdiction finds any provision, or portion thereof, to be unenforceable, the remainder of this Agreement shall continue in full force and effect. This Agreement shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods, the application of which is hereby expressly excluded.

7. **Termination.** Your rights under this Agreement will terminate automatically without notice from UCAR if you fail to comply with any term(s) of this Agreement. You may terminate this Agreement at any time by destroying the Software and any related documentation and any complete or partial copies thereof. Upon termination, all rights granted under this Agreement shall terminate. The following provisions shall survive termination: Sections 2, 3, 6 and 9.

8. **Complete Agreement.** This Agreement constitutes the entire agreement between the parties with respect to the use of the Software and supersedes all prior or contemporaneous understandings regarding such subject matter. No amendment to or modification of this Agreement will be binding unless in writing and signed by UCAR.

9. **Notices and Additional Terms.** Copyright in Software is held by UCAR. You must include, with each copy of the Software and associated documentation, a copy of this Agreement and the following notice:

   "The source of this material is the Research Applications Laboratory at the National Center for Atmospheric Research, a program of the University Corporation for Atmospheric Research (UCAR) pursuant to a Cooperative Agreement with the National Science Foundation; ©2007 University Corporation for Atmospheric Research. All Rights Reserved."

   The following notice shall be displayed on any scholarly works associated with, related to or derived from the Software:

   "Model Evaluation Tools (MET) was developed at the National Center for Atmospheric Research (NCAR) through a grant from the United States Air Force Weather Agency (AFWA). NCAR is sponsored by the United States National Science Foundation."

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

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. We are grateful to the individuals who participated in MET planning workshops in February 2007, April 2008, August 2009, and November 2010; the ideas generated at those workshops will help MET grow in future years. Finally, we would like to specifically thank the verification advisory group (Mike Baldwin, Matthew Sittel, Elizabeth Ebert, Geoff DiMego, Chris Davis, and Jason Knievel) 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. This chapter also documents the new Ensemble preprocessing tool. Chapters 4 through 7 focus on the main modules contained in the current version of MET, including the Point-Stat, Grid-Stat, MODE and Wavelet-Stat 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 8 and 9 focus on the analysis modules, Stat-Analysis and MODE-Analysis, which aggregate the output statistics from the other tools across multiple cases. Finally, Chapters 10 and 11 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 is posted on the MET website (http://www.dtcenter.org/met/users/). Note that the MET development group also accepts contributed analysis and plotting scripts which may be posted on the MET website for use by the community.

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

1.2 The Developmental Testbed Center (DTC)

MET has been developed, and will be maintained and enhanced, by the Developmental Testbed Center (DTC; http://www.dtcenter.org/). The main goal of the DTC is to serve

as a bridge between operations and research, to facilitate the activities of these two important components of the numerical weather prediction (NWP) community. The DTC provides an environment that is functionally equivalent to the operational environment in which the research community can test model enhancements; the operational community benefits from DTC testing and evaluation of models before new models are implemented operationally. MET serves both the research and operational communities in this way – offering capabilities for researchers to test their own enhancements to models and providing a capability for the DTC to evaluate the strengths and weaknesses of advances in NWP prior to operational implementation.

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

1.3 MET goals and design philosophy

The primary goal of MET development is to provide a state-of-the-art verification package to the NWP community. By "state-of-the-art" we mean that MET will incorporate newly developed and advanced verification methodologies, including new methods for diagnostic and spatial verification and new techniques provided by the verification and modeling communities. MET also utilizes and replicates the capabilities of existing systems for verification of NWP forecasts. For example, the MET package replicates existing 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 is maintained by the DTC in Boulder, Colorado. The MET package is freely available to the modeling, verification, and operational communities, including universities, governments, the private sector, and operational modeling and prediction centers.

1.4 MET components

The major components of the MET package are represented in Figure 1-1. The main stages represented are input, reformatting, 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.

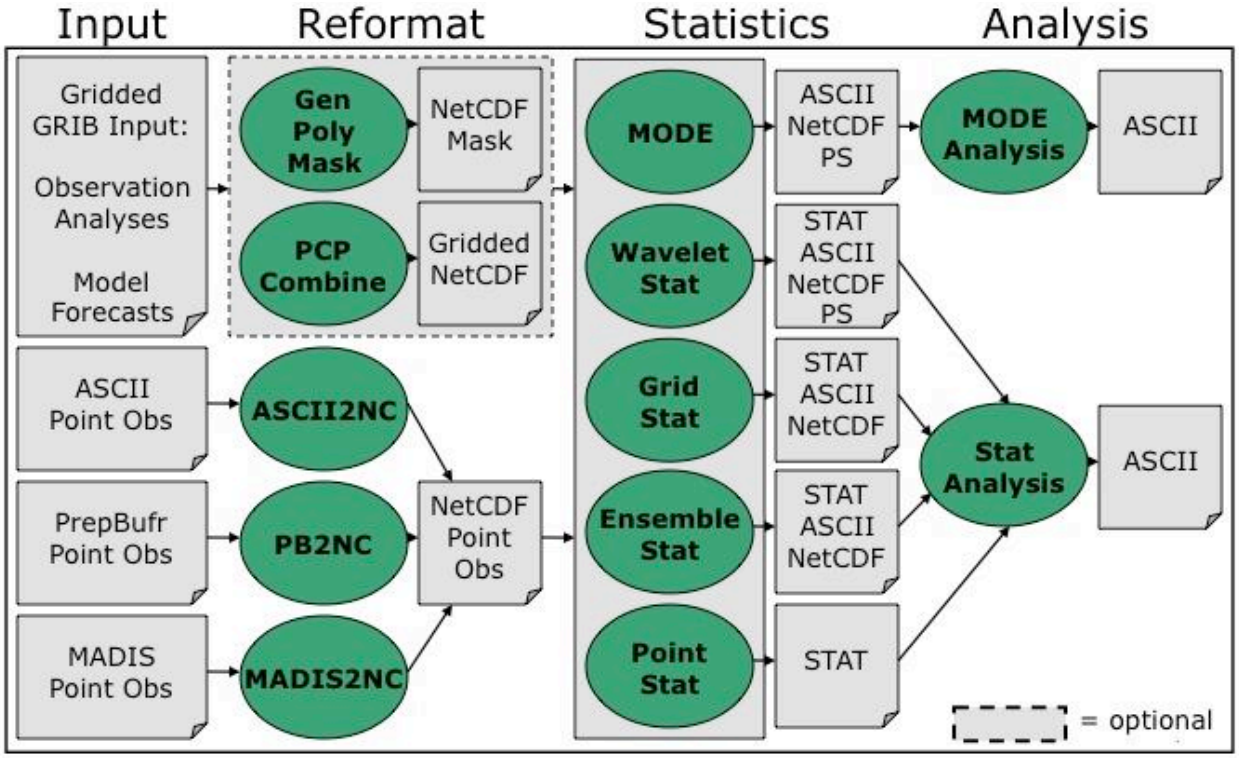


***Figure 1-1**. Basic representation of current MET structure and modules. Green areas represent software and modules included in MET, and gray areas represent input and output files.*

The reformatting stage of MET consists of the Gen-Poly-Mask, PB2NC, ASCII2NC, Pcp-Combine, and Ensemble Stat tools. The PB2NC tool is used to create NetCDF files from input PrepBufr files containing point observations. Likewise, the ASCII2NC tool is used to create NetCDF files from input ASCII point observations. METAR and RAOB data from the MADIS network can be formatted for use in MET by the madis2nc tool. These NetCDF files are then used in the statistical analysis step. The Gen-Poly-Mask and Pcp-Combine are optional. The Gen-Poly-Mask tool will create a bitmapped masking area from a user specified polygon, i.e. a text file containing a series of latitudes / longitudes. This mask can then be used to efficiently limit verification to the interior of a user specified region. The Pcp-Combine tool accumulates precipitation

amounts into the time interval selected by the user – if a user would like to verify over a different time interval than is included in their forecast or observational dataset. The Ensemble-Stat tool will combine many forecasts into an ensemble mean or probability forecast. Additionally, if observations are included ensemble rank histogram information is produced.

The four main statistical analysis components of the current version of MET are: Point-Stat, Grid-Stat, MODE, and Wavelet-Stat. 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. These intervals provide more meaningful information about forecast performance. For example, confidence intervals allow credible comparisons of performance between two models when a limited number of model runs is available.

Sometimes it may be useful to verify a forecast against gridded fields (e.g., Stage IV precipitation analyses). The Grid-Stat tool produces traditional verification statistics when a gridded field is used as the observational dataset. Like the Point-Stat tool, the Grid-Stat tool also produces confidence intervals. The Grid-Stat tool also 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.

The Wavelet-Stat tool decomposes two-dimensional forecasts and observations according to the Intensity-Scale verification technique described by Casati et al. (2004). There are many types of spatial verification approaches and the Intensity-Scale technique belongs to the scale-decomposition (or scale-separation) verification approaches. The spatial scale components are obtained by applying a wavelet transformation to the forecast and observation fields. The resulting scale-decomposition

measures error, bias and skill of the forecast on each spatial scale. Information is provided on the scale dependency of the error and skill, on the no-skill to skill transition scale, and on the ability of the forecast to reproduce the observed scale structure. The Wavelet-Stat tool is primarily used for precipitation fields. However, the tool can be applied to other variables, such as cloud fraction.

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

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

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
- Hurricane track verification
- Enhanced support for wind direction verification

Support for other input formats

- Support for gridded data in NetCDF, CF convention

Additional analysis capabilities and plotting routines

- Post to the MET website sample analysis and plotting routines that may include
    - Boxplots
    - Discrimination plots

- Reliability diagrams
- Scatter/density plots
- Color-fill/contour maps of statistics
- Height series
- Histograms

Other capabilities

- Autoconf configurability
- Database and display system for the statistical output of MET

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	Linux	GNU / PGI / Intel
IBM	Power Series	AIX	IBM

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.

2.4 Required compilers and scripting languages

The MET package was developed and tested using the GNU `g++/gfortran` 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 be included as command line utilities.

In order to control the desired flow through MET, users are encouraged to run the tools via a script (see Chapter 10 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** may be required depending on which Fortran compiler is used to compile MET. It is not necessary when using the GNU gfortran and PGI pgf77 compilers but is required for the GNU g77 compiler. 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.

Two additional utilities are strongly recommended for use with MET:

1. The **Unified Post-Processor** is recommended for post-processing the raw model output prior to verifying the model forecasts with MET. The Unified 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 Unified Post-Processor outputs model data in this format from both WRF cores, the NMM and the ARW. However, the Unified Post-Processor is not strictly required as long as the user can produce 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 the Grid-Stat, Wavelet-Stat, or MODE tools. Prior to running MET, the model output and verifying gridded observations must be placed on the same grid. The copygb utility is distributed as part of the Unified 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.

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 C and Fortran 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 compilers consists of the following 3 steps:

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

Alternatively, compiling the BUFRLIB using the PGI C and Fortran-77 compilers consists of the following 3 steps:

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

Compiling the BUFRLIB using the IBM C and Fortran compilers consists of the following 3 steps:

- **`xlc -c -DUNDERSCORE *.c`**
- **`xlf -c -qextname *.f *.F`**
- **`ar crv libbufr.a *.o`**

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, two additional utilities are strongly recommended for use with MET.

1. The **Unified Post-Processor** is recommended for post-processing the raw model output prior to verifying the data with MET. The Unified 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 Unified Post-Processor. Installation instructions for the Unified Post-Processor can be found in Chapter 7 of the WRF-NMM User's Guide or online at http://www.dtcenter.org/wrf-nmm/users/docs/user_guide/V3/users_guide_nmm_chap7.pdf .

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

2.8 MET directory structure

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

- **METv4.0/**
  - **bin/**
  - **data/**
  - **doc/**
  - **export.mk**
  - **include/**
  - **lib/**
  - **Makefile**
  - **met_defs.mk**
  - **out/**
  - **README**
  - **scripts/**
  - **src/**
  - **user_defs_gnu.mk**
  - **user_defs_ibm.mk**
  - **user_defs_intel.mk**
  - **user_defs.mk**
  - **user_defs_pgi.mk**

The top-level MET directory consists of a README file, 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`**, **`ensemble_stat`**, **`pb2nc`**, **`ascii2nc`**, **`madis2nc`**, **`wwmca_regrid`**, **`gen_poly_mask`**, **`pcp_combine`**, **`point_stat`**, **`stat_analysis`**, **`wavelet_stat`**) as well as some plotting utilities.

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.

Install the required libraries.

- Please refer to Section 2.6 on how to install the required libraries.

Configure the top-level user_defs.mk

- Once you have downloaded the MET tarball, unzip and unpack its contents (refer to Section 2.8).
- Make a copy of the user_defs.mk most similar to your OS and compiler. For example, if compiling on Linux using the GNU compilers:
- cp user_defs_gnu.mk user_defs.mk
- Edit the top-level user_defs.mk 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.
    - If required for your compiler, set F2C_BASE to the location where the F2C or G2C library is installed if it is not installed in a standard location.
    - If required for your compiler, set F2C_LIBNAME to either –lf2c or –lg2c to indicate which library is to be used.
    - Set the GRIB2C_BASE, GRIB2C_INCS, and GRIB2C_LIBS to the correct paths if you will be using files in GRIB2 format. Also, make sure the WITH_GRIB2 flag is set to 1, so the GRIB2 portions of the code will be compiled.
    - The additional parameters in the user_defs.mk may be set as needed to configure the build to your system such as compiler flags and additional libraries.

Execute the build

- Execute the GNU make command, typically by typing make, to build the MET package. Note that on IBM machines, the GNU make command may be named gmake:
- make>& 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 *****

- ² Several compilation warnings may occur which are expected.
- ² 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: All of these test scripts should take less than 10 minutes to run on most machines.

- 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.

- 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 Unified Post-Processor). Point observation files may be supplied in either PrepBufr, ASCII, or MADIS format. Note that MET does not require the Unified Post-Processor to be used, but does require that the input GRIB data be on a standard, de-staggered grid on pressure or regular levels in the vertical. While the Grid-Stat, Wavelet-Stat, and MODE tools can be run on a gridded field at virtually any level, the Point-Stat tool can only be used to verify forecasts at the surface or on pressure levels.

When comparing two gridded fields with the Grid-Stat, Wavelet-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 wish to use observations not included in the standard PrepBufr files. For this reason, prior to performing the verification step in the Point-Stat tool, the PrepBufr file is reformatted with the PB2NC tool. In this step, the user can select various ways of stratifying the observation data spatially, temporally, and by type. The remaining observations are reformatted into an intermediate NetCDF file. The ASCII2NC tool may be used to convert ASCII point observations that are not available in the PrepBufr files into this NetCDF format for use by the Point-Stat verification tool. Users with METAR or RAOB data from MADIS can convert these observations into NetCDF format with the new madis2nc tool, then use them with the Point-Stat verification tool.

3.2 Intermediate data formats

MET uses NetCDF as an intermediate file format. The Ensemble-Tool, WWMCA-Tool, Pcp-Combine, Gen-Poly-Mask, 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 accumulated precipitation in two GRIB files directly. The command line arguments for the Pcp-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 Gen-Poly-Mask tool is used to define a bitmapped masking region that can be used by the Ensemble-Tool, Grid-Stat, Point-Stat, and MODE as a verification subdomain. It is generally more efficient to use the NetCDF output of `gen_poly_mask` to define a masking region than using a complex polyline directly in the other MET tools. However, the NetCDF output can only be applied to datasets on a common domain. It must be regenerated for each domain used.

The PB2NC tool is used to reformat the input PrepBufr files containing point observations. This tool 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: STAT files, ASCII files, NetCDF files, and PostScript plots.

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

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

The MODE tool creates two ASCII output files as well (although they are not in a STAT format) and also 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.

The Ensemble-Tool, Grid-Stat, Wavelet-Stat, and 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 cluster object fields. The Ensemble-Tool creates a NetCDF file containing the ensemble forecast values, statistics, and, if requested, matched observations for each verification region and variable type/level requested in the configuration file. In addition, when rank histogram information is requested, the NetCDF file contains the observation rank values. 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. The Wavelet-Stat tool creates a NetCDF file summarizing the wavelet decomposition of the forecast and observation fields for each variable type/level, raw threshold, and tile masking region chosen. The generation of these files is controlled by configuration files or command line switches. As discussed in the previous section, the Pcp-Combine and Gen-Poly-Mask tools create gridded NetCDF output as well, while the PB2NC and ASCII2NC tools create intermediate NetCDF files containing point observations.

The MODE and Wavelet-Stat tools produce PostScript plots summarizing the features-based approach used in the verification. The PostScript plots are generated using internal libraries and do not depend on an external plotting package. The MODE plots contain several summary pages at the beginning, but the total number of pages will depend on the merging options chosen. Additional pages will be created if merging is

performed using the double thresholding or fuzzy engine merging techniques for the forecast and observation fields. The number of pages in the Wavelet-Stat plots depend on the number of masking tiles used and the dimension of those tiles. The first summary page is followed by plots for the wavelet decomposition of the forecast and observation fields. The generation of these PostScript output files can be disabled using command line options.

3.4 Data format summary

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

1. **PB2NC Tool**
   - **Input**: One PrepBufr point observation file and one configuration file.
   - **Output**: One NetCDF file containing the observations that have been retained.

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

3. **MADIS2NC Tool**
   - **Input**: One MADIS point observation file.
   - **Output**: One NetCDF file containing the reformatted observations.

4. **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.

5. **Gen-Poly-Mask Tool**
   - **Input**: One gridded model or observation file in GRIB1 format and one ASCII file defining a Lat/Lon masking polyline.
   - **Output**: One NetCDF file containing a bitmap for the masking region defined by the polyline over the domain of the gridded input file.

6. **Ensemble Stat Tool**
   - **Input**: An arbitrary number of gridded model files in GRIB1 format and one or more optional files containing observations. The observations may be in either netCDF or GRIB1 format. Point and gridded observations are both accepted.
   - **Output**: One NetCDF file containing requested ensemble forecast information and, where applicable, rank histogram information.

7. **Point-Stat Tool**
   - **Input**: One model file either in GRIB1 format or in the NetCDF format output from the Pcp-Combine tool, at least one point observation file in NetCDF format (as the output of the PB2NC or ASCII2NC tool), and one configuration file.
   - **Output**: One STAT file containing all of the requested line types, and several ASCII files for each line type requested.

8. **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 STAT file containing all of the requested line types, several ASCII files for each line type requested, and one NetCDF file containing the matched pair data and difference field for each verification region and variable type/level being verified.

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

10. **Wavelet-Stat Tool**
    - **Input**: One model file and one gridded observation file either in GRIB1 format or in the NetCDF format output from the Pcp-Combine tool, and one configuration file.
    - **Output**: One STAT file containing the 'ISC" line type, one ASCII file containing intensity-scale information and statistics, one NetCDF file containing information about the wavelet decomposition of forecast and

observed fields and their differences, and one PostScript file containing plots and summaries of the intensity-scale verification.

11. **Stat-Analysis Tool**
    - **Input**: One or more STAT 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 STAT data.
    - **Output**: ASCII output of the analysis jobs will be printed to the screen unless redirected to a file using the "-out" option.

12. **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.

Please note that in earlier version of the PB2NC tool, users were required to run their PrepBufr files through the cwordsh tool to perform Fortran-blocking on their PrepBufr files prior to running them through PB2NC. That step is no longer required since the Fortran-blocking is now done internally.

3.5.1 `pb2nc` *usage*

The usage statement for the PB2NC tool is shown below:

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

```
[-dump path]
[-log file]
[-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 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_beg time** option in YYYYMMDD[_HH[MMSS]] format sets the beginning of the retention time window.

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

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

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

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

7. 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 ADPUPA and ADPSFC message types might consist of the following:

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

The example script for running MODE included in section 10.2 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 = -5400;
end =  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** and **end** variables define a time window around the file's observation time for PrepBufr messages that should be retained. **beg** 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** 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 = "";
   poly = "";
};
```

The **grid** and **poly** variables are used to define a spatial masking region for retaining observations. **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. **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`.

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

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

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

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

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

level_category = [];
```

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

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

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_thresh = 2;

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

---

event_stack_flag = 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.

---

tmp_dir = “/tmp”;

The **tmp_dir** indicates where temporary files should be written.

---

version = “V4.0”;

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 (16)
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 releases 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 observation location.
6	Elevation	Elevation in msl of the observing location.
7	Grib_Code	Integer 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 is 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]
        [-log file]
        [-v level]
```

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

***Required arguments for* `ascii2nc`**

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`**

3. 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.

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

5. The **`-v` level** option indicates the desired level of verbosity. The 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 MADIS2NC tool

This section describes how to run the MADIS2NC tool. The MADIS2NC tool is used to reformat Meteorological Assimilation Data Ingest System (MADIS) point observations into the NetCDF format expected by the Point-Stat tool. More information about MADIS data and formatting is available at **madis.noaa.gov**. Since the MADIS2NC tool simply performs a reformatting step, no configuration file is needed.

3.7.1 `madis2nc` *usage*

The usage statement for MADIS2NC tool is shown below:

```
Usage: madis2nc
     madis_file
     out_file
     [-type str]
     [-qc_dd list]
     [-lvl_dim list]
     [-rec_beg n]
     [-rec_end n]
     [-log file]
     [-v level]
```

`madis2nc` has two required arguments and can take optional ones.

Required arguments for madis2nc

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

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

***Optional arguments for* `madis2nc`**

3. Optional argument **-type str** specifies the type of MADIS observations (metar or raob).

4. The -**qc_dd list** option specifies a comma-separated list of QC flag values to be accepted (Z,C,S,V,X,Q,K,G,B).

5. The **-lvl_dim list option** specifies a comma-separated list of vertical level dimensions to be processed.

6. To specify the exact records to be processed, the **-rec_beg n** specifies the index of the first MADIS record to process and **-rec_end n** specifies the index of the last MADIS record to process. Both are zero-based.

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

8. The **-v level** option indicates the desired level of verbosity. The value of "level" will override the default setting of 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 `madis2nc` calling sequence is shown below:

```
madis2nc  sample_madis_obs.txt
          sample_ madis _obs.nc –log madislog –v2
```

In this example, the MADIS2NC tool will reformat the input **sample_madis_obs.txt** file into NetCDF format and write the output to a file named **sample_madis_obs.nc**. Warnings and error messages will be written to the madislog file, and the verbosity will be level two.

3.8 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 MET statistics tools. Use of Pcp-Combine on a single file will result in that file being written out in netCDF format, with no changes to the content. 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.

3.8.1 `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]
     [-ptv number]
     [-log file]
     [-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 arguments for the **add**:

```
ADD_ARGS:
         in_file1
         accum1
         [in_file2 accum2 in_file3 accum3 . . . ]
         out_file
```

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

```
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 **-ptv number** option may be used to specify which GRIB parameter table version number should be used for interpreting the meaning of GRIB codes.
3. The **-log file** option directs output and errors to the specified log file. All messages will be written to that file as well as cout and cerr. Thus, users can

save the messages without having to redirect the output on the command line. The default behavior is no logfile.

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

Required arguments for the `pcp_combine` sum command

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

2. The **in_accum** argument, provided in HH[MMSS] format, indicates the accumulation interval of the model or observation 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[_HH[MMSS]] format, indicates the desired valid time to which the accumulated precipitation is to be summed.

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

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 directories 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 command

1. The **in_file1** argument indicates the first GRIB file to be processed.

2. The **in_accum1** argument, provided in HH[MMSS] format, indicates the accumulation interval to be extracted from the first GRIB file.

An arbitrary number of additional files and accumulations can be provided. All of them will be added and the total will be placed in the output file.

Required arguments for the* `pcp_combine` *subtract command

1. The **in_file1** argument indicates the first GRIB file to be processed.

2. The **in_accum1** argument, provided in HH[MMSS] 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[MMSS] format, indicates the accumulation interval to be extracted from the second GRIB file. This 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.

3.8.2 `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, MODE, and Wavelet 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.

3.9 Gen-Poly-Mask tool

This section contains a description of running the Gen-Poly-Mask tool. The Gen-Poly-Mask tool may be run to create a bitmap verification masking region to be used by the Grid-Stat, Point-Stat, and MODE tools. This tool enables the user to generate a polyline masking region once for a domain and apply it to many cases. When using a complex polyline containing hundreds of vertices, it is a good idea to use the Gen-Poly-Mask tool to create a bitmap masking region before running the Grid-Stat, Point-Stat, and MODE tools. Doing so will make the Grid-Stat, Point-Stat, and MODE tools run more efficiently.

3.9.1 `gen_poly_mask` *usage*

The usage statement for the Gen-Poly-Mask tool is shown below:

```
Usage: gen_poly_mask
        data_file
        mask_poly
        netcdf_file
        [-rec i]
        [-v level]
```

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

Required arguments for **`gen_poly_mask`**

1. The **data_file** argument indicates the name of a GRIB1 or the NetCDF output of the Pcp-Combine tool which defines the domain over which the masking bitmap is to be defined.

2. The **mask_poly** argument indicates the name of the ASCII Lat/Lon polyline file defining the masking region.

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

Optional arguments for **`gen_poly_mask`**

1. The **`–rec i`** optional argument can be used to specify which GRIB record is used to define the domain. By default, the domain information will be extracted from the first record. This argument would only be used in the case of a single GRIB file which contains records defined on different domains.

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 `gen_poly_mask` calling sequence is shown below:

```
gen_poly_mask  sample_fcst.grb
               CONUS.poly
               CONUS_poly.nc
```

In this example, the Gen-Poly-Mask tool will apply the polyline defined in the file **`CONUS.poly`** to the domain on which the data in the file **`sample_fcst.grb`** resides. It will create a NetCDF file containing a bitmap for the domain with a value of 1 for all grid points inside the CONUS polyline and a value of 0 for all grid points outside. It will write an output NetCDF file named **`CONUS_poly.nc`**.

3.10 Ensemble Stat tool

This section contains a description of running the Ensemble Stat tool. This tool may be run to create ensemble forecasts (mean, probability, spread, etc) from a set of several forecast model files to be used by the MET statistics tools. If observations are also included, ensemble statistics such as rank histograms and continuous ranked probability score are produced.

3.10.1 **`ensemble_stat`** *usage*

The usage statement for the Ensemble Stat tool is shown below:

```
Usage: ensemble_stat
       n_ens ens_file_1 ... ens_file_n | ens_file_list
       config_file
       [-grid_obs file]
       [-point_obs file]
       [-ens_valid time]
       [-ens_lead time]
       [-obs_valid_beg time]
       [-obs_valid_end time]
       [-obs_lead time]
       [-outdir path]
       [-v level]
```

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

Required arguments ensemble_stat

1. The "n_ens ens_file_1 ... ens_file_n" is the number of ensemble members followed by a list of ensemble member file names. This argument is not required when ensemble files are specified in the “ens_file_list”, detailed below.
2. The "ens_file_list" is an ASCII file containing a list of ensemble member file names. This is not required when a file list is included on the command line, as in #2 above.
3. The "config_file" is an EnsembleStatConfig file containing the desired configuration settings.

NOTE: The ensemble members and gridded observations must be on the same grid.

***Optional arguments for* `ensemble_stat`**

1. To produce rank histogram information when you have gridded observations, use the "-grid_obs file" option to specify a gridded observation file. This option may be used multiple times if your observations are in several files.

2. To produce rank histogram information when you have point observations, use the "-point_obs file" to specify a NetCDF point observation file. This option may be used multiple times if your observations are in several files.

3. The optional "-ens_valid time" in YYYYMMDD[_HH[MMSS]] format sets the ensemble valid time to be used.

4. Setting "-ens_lead time" in HH[MMSS] format sets the ensemble lead time to be used (optional).

5. To filter observations by time, use "-obs_valid_beg time" in YYYYMMDD[_HH[MMSS]] format to set the beginning of the matching observation time window.

6. As above, use "-obs_valid_end time" in YYYYMMDD[_HH[MMSS]] format to set the end of the matching observation time window.

7. Set "-obs_lead time" in HH[MMSS] format to choose the observation lead time to be used.

8. Specify the "-outdir path" option to override the default output directory (/out/ensemble_stat).

9. 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 `ensemble_stat` calling sequence is shown below:

```
ensemble_stat \
   6 sample_fcst/2009123112/*gep*/d01_2009123112_02400.grib \
   config/EnsembleStatConfig \
   -grid_obs sample_obs/ST4/ST4.2010010112.24h \
   -point_obs out/ascii2nc/precip24_2010010112.nc \
   -outdir out/ensemble_stat -v 2
```

In this example, the Ensemble Stat tool will process six forecast files specified in the file list into an ensemble forecast. Observations in both point and grid format will be included, and used to calculate ranks separately. These ranks can then be used to plot a rank histogram for each type of observation. Ensemble Stat will create a NetCDF file containing requested ensemble fields and an ascii stat file.

3.10.2 `ensemble_stat` *output*

The ensemble_stat tool can calculate any of the following:

Ensemble Mean fields
Ensemble Standard Deviation fields
Ensemble Mean - 1 Standard Deviation fields
Ensemble Mean + 1 Standard Deviation fields
Ensemble Minimum fields
Ensemble Maximum fields
Ensemble Range fields
Ensemble Valid Data Count fields
Ensemble Relative Frequency by threshold fields
Ranked Histograms (if Observation Field Provided)

The ensemble_stat tool then writes:

Gridded fields of Ensemble forecast values to a NetCDF file
Gridded field of Observation Ranks to a NetCDF file
Stat file with Rank Histogram and Ensemble information
Observation Rank Matched Pairs

Table 3-5. *Header information for ensemble-stat output RHIST file.*

HEADER		
Column Number	**Header Column Name**	**Description**
1	VERSION	Version number (set to 3.0)
2	MODEL	User provided text string designating model name
3	FCST_LEAD	Forecast lead time in HHMMSS format

HEADER		
Column Number	**Header Column Name**	**Description**
4	FCST_VALID_BEG	Forecast valid start time in YYYYMMDDHH format
5	FCST_VALID_END	Forecast valid end time in YYYYMMDDHH format
6	OBS_LEAD	Observation lead time in HHMMSS format
7	OBS_VALID_BEG	Observation valid start time in YYYYMMDDHH format
8	OBS_VALID_END	Observation valid end time in YYYYMMDDHH format
9	FCST_VAR	Model variable
10	FCST_LEV	Selected Vertical level for forecast
11	OBS_VAR	Observed variable
12	OBS_LEV	Selected Vertical level for observations
13	OBTYPE	Type of observation selected
14	VX_MASK	Verifying masking region indicating the masking grid or polyline region applied
15	INTERP_MTHD	Interpolation method applied to forecasts
16	INTERP_PNTS	Number of points used in interpolation method
17	FCST_THRESH	The threshold applied to the forecast
18	OBS_THRESH	The threshold applied to the observations
19	COV_THRESH	NA in Point-Stat
20	ALPHA	Error percent value used in confidence intervals
21	LINE_TYPE	RHIST
22	TOTAL	Count of observations
23	CRPS	Continuous Ranked Probability Score
24	IGN	Ignorance score
25	N_RANK	Number of possible ranks for observation
26-?	RANK_?	# of instances that observation has this rank

Table 3-6. *Header information for ensemble-stat output ORANK file.*

Column Number	**Header Column Name**	**Description**
1-21		Same as in Table 3-5 above.
22	TOTAL	Count of observations
23	INDEX	Line number in ORANK file
24	OBS_SID	Station Identifiier
25	OBS_LAT	Latitude of the observation
26	OBS_LON	Longitude of the observation
27	OBS_LVL	Level of the observation
28	OBS_ELV	Elevation of the observation
29	OBS	Value of the observation
30	PIT	Probability Integral Transform

Column Number	Header Column Name	Description
31	RANK	Rank of the observation
32	N_ENS_VLD	Number of valid ensemble values
33	N_ENS	Number of ensemble values
34-?	ENS_?	Value of each ensemble member

3.9.3 `ensemble_stat` *configuration file*

The default configuration file for the Ensemble-Stat tool named `EnsembleStatConfig_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 tool.

```
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 STAT output so that verification statistics from multiple models may be differentiated. The value listed above is simply set to "WRF".

```
ens = {
   ens_thresh = 1.0;
   vld_thresh = 1.0;

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

When summarizing the ensemble, compute a ratio of the number of valid ensemble fields to the total number of ensemble members. If this ratio is less than the

ens_thresh, then quit with an error. This threshold must be between 0 and 1. Setting this threshold to 1 will require that all ensemble members be present to be processed.

When summarizing the ensemble, for each grid point compute a ratio of the number of valid data values to the number of ensemble members. If that ratio is less than **vld_thresh,** write out bad data. This threshold must be between 0 and 1. Setting this threshold to 1 will require each grid point to contain valid data for all ensemble members.

For each **field** listed in the forecast field, give the name and vertical or accumulation level, plus one or more categorical thresholds. The thresholds are specified using 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. Symbols can also be used as shown above. It is the user's responsibility to know the units for each model variable and to choose appropriate threshold values. For probabilistic forecasts, thresholds must be specified using "ge" convention.

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

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

The **fcst** and **obs** variables should specify a list of ensemble field names and levels to be verified, with the corresponding observations (**message_type**).

The Ensemble-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.

The **obs** can be set equal to **fcst** to use the identical fields.

```
beg = -5400;
end =  5400;
```

Each gridded forecast file has a valid time associated with it. The **beg** and **end** variables define a time window in seconds around the valid time of the forecast file for the observations to be matched to it. For a forecast valid time, v, observations with a valid time falling in the time window [v+beg_ds, v+end_ds] will be used. These selections are overridden by the command line arguments **–valid_beg** and **–valid_end**.

```
mask = {
   grid = [ "FULL" ];
   poly = [];
   sid  = "";
};
```

The **mask** group contains information about the domain to be verified. It allows the user to place a 'mask' over their existing grid to hide some areas from the verification and only use the remaining areas.

The **grid** 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.

The **poly** variable contains a comma-separated list of files that define verification masking regions. These masking regions may be specified in two ways: as a lat/lon polygon or using a gridded data file such as the NetCDF output of the Gen-Poly-Mask tool.

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.

Alternatively, any gridded data file that MET can read may be used to define a verification masking region. Users must specify a description of the field to be used from the input file and, optionally, may specify a threshold to be applied to that field. Any grid point where the resulting field is 0, the mask is turned off. Any grid point where it is non-zero, the mask is turned on.

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

```
interp = {
   field      = BOTH;
   vld_thresh = 1.0;

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

The **interp** group controls interpolation of forecast and/or observation fields. Parameters for interpolation include the field to be interpolated, the minimum valid points to be included in the interpolation, the type of interpolation to be performed, and the width of the grid to use in the interpolation.

The **field** variable controls how the interpolation should be applied. A value of FCST will smooth, e.g. apply the interpolation, to only the forecast field. A value of OBS applies interpolation to only the observation field. To smooth both the forecast and observation fields, set the value to BOTH.

The **vld_thresh** 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.

The **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, LS_FIT, and BILIN for the minimum, maximum, median, unweighted mean, distance-weighted mean, least squares fit, and bilinear interpolation. 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.

The **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 **width** value specifies the

width of that square. A **width** value of 1 is interpreted as the nearest neighbor model grid point to the observation point. A **width** of 2 defines a 2 x 2 square of grid points around the observation point (the 4 closest model grid points), while a **width** 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.

```
output_flag = {
   rhist = BOTH;
   orank = BOTH;
};
```

The **output_flag** array controls the type of output that the Point-Stat tool generates. Each flag corresponds to an output line type in the STAT file. Setting the flag to NONE indicates that the line type should not be generated. Setting the flag to STAT indicates that the line type should be written to the STAT file only. Setting the flag to BOTH indicates that the line type should be written to the STAT file as well as a separate ASCII file where the data is grouped by line type. Note that these two line types are easily derived from each other. 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.

```
ensemble_flag = {
   mean      = TRUE;
   stdev     = TRUE;
   minus     = TRUE;
   plus      = TRUE;
   min       = TRUE;
   max       = TRUE;
   range     = TRUE;
   vld_count = TRUE;
   frequency = TRUE;
   rank      = TRUE;
};
```

The **ensemble_flag** specifies which derived ensemble fields should be calculated and output. Setting the flag to TRUE produces output of the specified field, FALSE produces no output for that field type. The flags correspond to the following output line types:

1. Ensemble Mean Field
2. Ensemble Standard Deviation Field
3. Ensemble Mean – One Standard Deviation Field
4. Ensemble Mean + One Standard Deviation Field
5. Ensemble Minimum Field
6. Ensemble Maximum Field
7. Ensemble Range Field

8. Ensemble Valid Data Count
9. Ensemble Relative Frequency by threshold Fields
10. Gridded Field of Observation Ranks written to netCDF file

---

```
rng = {
   type = "mt19937";
   seed = "";
}
```

The **rng** group defines the random number generator **type** and **seed** 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.

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

```
duplicate_flag = NONE;
```

The reporting mechanism for this feature can be activated by specifying a verbosity level of three or higher. The report will show information about where duplicates were detected and which observations were used in those cases.

```
output_prefix = "";
```

This option specifies a string to be used in the output file name. It can be useful for keeping results for different models or variables from overwriting each other.

```
version = "v4.0";
```

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.

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 and probabilistic statistics generally are derived by applying a threshold to the forecast and observation values. Confidence intervals – representing the uncertainty in the verification measures – are computed for the verification statistics.

Scientific and statistical aspects of the Point-Stat tool are discussed in the following section. Practical aspects of the Point-Stat tool are described in 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 grid point 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. Matching in the vertical and horizontal are completed separately using different methods.

In the vertical, if forecasts and observations are at the same vertical level, then they are paired as is. If any discrepancy exists between the vertical levels, then the forecasts are interpolated to the level of the observation. The vertical interpolation is done in natural log of pressure coordinates, except for specific humidity, which is interpolated using the natural log of specific humidity in natural log of pressure coordinates. When forecasts are for the surface, no interpolation is done. They are matched to observations with message type ADPSFC or SFCSHP.

To match forecasts and observations in the horizontal plane, the user can select from a number of methods described below. Many of these methods require the user to define the width of the forecast grid *W*, around each observation point *P,* that should be considered. 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 consisting of

9 grid points centered on the grid point closest to *P*. Fig. 4-1 provides illustration. The point *P* denotes the observation location where the interpolated value is calculated. The interpolation width *W*, shown is five.

This section describes the options for interpolation in the horizontal.

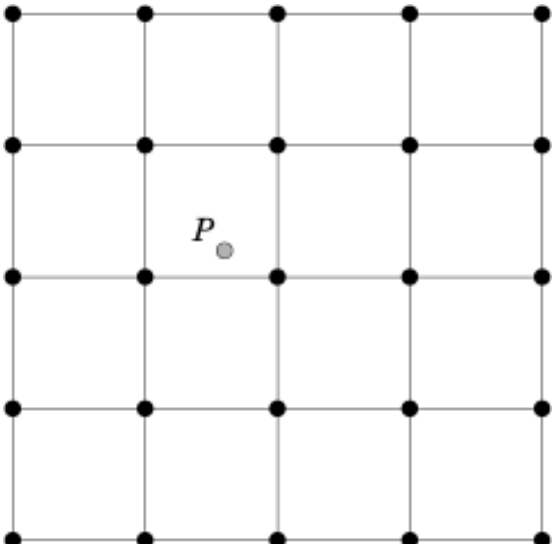


***Figure 4-1**: Diagram illustrating matching and interpolation methods used in MET. See text for explanation.*

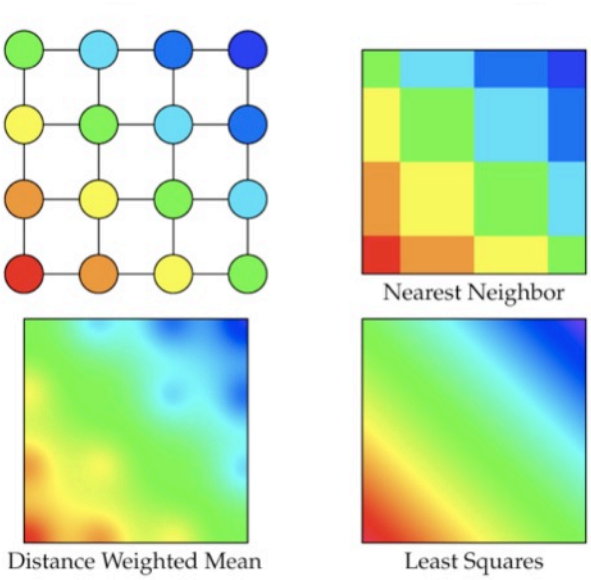


***Figure 4-2**: Illustration of some matching and interpolation methods used in MET. See text for explanation.*

Nearest Neighbor

The forecast value at *P* is assigned the value at the nearest grid point. No interpolation is performed. Here, "nearest" means spatially closest in horizontal grid coordinates. This method is used by default when the interpolation width, W, is set to 1.

Minimum value

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

Maximum value

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

Distance-weighted mean

The forecast value at *P* is a weighted sum of the values in the W x W square. The weight given to each forecast point is the reciprocal of the square of the distance (in grid coordinates) from *P*. The weighted sum of forecast values is normalized by dividing by the sum of the weights.

Unweighted mean

This method is similar to the distance-weighted mean, except all the weights are equal to 1. The distance of any point from *P* is not considered.

Median

The forecast value at *P* is the median of the forecast values in the 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*.

Bilinear Interpolation

This method is performed using the four closest grid squares. The forecast values are interpolated linearly first in one dimension and then the other to the location of the observation.

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 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 < 273.15°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.

Measures for probabilistic forecasts and dichotomous outcomes
For probabilistic forecasts, many verification measures are based on reliability, accuracy and bias. However, it also is of interest to investigate joint and conditional distributions of the forecasts and the observations, as in Wilks (2006). See Appendix C for specific information.

Probabilistic forecast values are assumed to have a range of either 0 to 1 or 0 to 100. If the max data value is > 1, we assume the data range is 0 to 100, and divide all the values by 100. If the max data value is <= 1, then we use the values as is. Further,

thresholds are applied to the probabilities with equality on the lower end. For example, with a forecast probability *p*, and thresholds *t1* and t*2*, the range is defined as: *t1* <= *p* < *t2*. The exception is for the highest set of thresholds, when the range includes 1: *t1* <= *p* <= 1.

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-α)% 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-α)% 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 each replicate. That is, given a

set of *n* forecast-observation pairs, we draw values at random from these pairs, allowing the same pair to be drawn more than once, and the statistic(s) is (are) calculated for each replicated sample. This yields a sample of the statistic(s) based solely on the data without making any assumptions about the underlying distribution of the sample. It should be noted, however, that if the observed sample of matched pairs is dependent, then this dependence should be taken into account somehow. Currently, in the confidence interval methods in MET do not take into account dependence, but future releases will support a robust method allowing for dependence in the original sample. More detailed information about the bootstrap algorithm is found in the appendix.

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

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

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

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

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

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

MET provides parametric confidence intervals based on assumptions of normality for the following probabilistic statistics:

- Brier Score

MET 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 statistics would be computationally unrealistic.

For more information on confidence intervals pertaining to verification measures, see Wilks (2006), Jolliffe and Stephenson (2003), and Bradley (2008).

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]
        [-point_obs netcdf_observation_file]
        [-obs_valid_beg time]
        [-obs_valid_end time]
        [-outdir path]
        [-log file]
        [-v level]
```

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

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

1. The **fcst_file** argument names the GRIB file or the NetCDF output of pcp_combine 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 **`–point_obs` netcdf_file** may be used to pass additional NetCDF point observation files to be used in the verification.

3. The **`–obs_valid_beg` time** option in YYYYMMDD[_HH[MMSS]] format sets the beginning of the observation matching time window.

4. The **`–obs_valid_end` time** option in YYYYMMDD[_HH[MMSS]] format sets the end of the observation matching time window.

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

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

7. 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. **A web tool that generates config file text is available via a link from the MET Users' web site.** This tool contains reasonable defaults for most fields. 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 tool.

```
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 STAT output so that verification statistics from multiple models may be differentiated. The value listed above is simply set to "WRF".

```
fcst = {
   wind_thresh  = [ NA ];
   message_type = [ "ADPUPA" ];

   field = [
      {
        name       = "SPFH";
        level      = [ "P500" ];
        cat_thresh = [ >80.0 ];
      },

      {
        name       = "TMP";
        level      = [ "P500" ];
        cat_thresh = [ >273.0 ];
      },

      {
```

```
      name       = "HGT";
      level      = [ "P500" ];
      cat_thresh = [ >0.0 ];
    },

    {
      name       = "UGRD";
      level      = [ "P500" ];
      cat_thresh = [ >5.0 ];
    },

    {
      name       = "VGRD";
      level      = [ "P500" ];
      cat_thresh = [ >5.0 ];
    }
  ];

};
obs = fcst;
```

or
`fcst_field[] = [ "RAINC(0,*,*)", "QVAPOR(0,5,*,*)" ];` for NetCDF input

The **fcst** group contains information about the model forecast fields, levels and threshold to use in verification. The **field** variable contains a comma-separated list of model variables and corresponding vertical levels to be verified. Each field is specified as a GRIB code or abbreviation followed by an accumulation or vertical level indicator for GRIB files or as a variable name followed by a list of dimensions for NetCDF files output from p_interp or MET.

For GRIB files, 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**", "**PNNN-NNN**", "**LNNN**", or "**RNNN**" must correspond to each field name. These indicate an accumulation interval, a single vertical level, a single pressure level, a range of pressure levels, a generic level, and a specific GRIB record number, respectively. "**NNN**" indicates the accumulation or level value.

To specify verification fields for NetCDF files, the name of the NetCDF variable should appear in the name filed. The level field should be of the form (i,...,j,*,*) for a single field. Here, i,...,j specifies fixed dimension values, and *,* specifies the two dimensions for the gridded field.

The values listed above indicate that specific humidity, temperature, height, wind speed, 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. Wind speed is typically not available in model files, but if it is, it will be used directly. However, if wind speed is unavailable but the U and V components of the wind are included, the wind speed will be calculated automatically by MET, provided that WIND is included in the **field** group.

For each **name** listed in the forecast field, 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. Multiple thresholds may be applied to each field code by providing a space-separated list within the double quotes (e.g. "gt0 le0"). It is the user's responsibility to know the units for each model variable and to choose appropriate threshold values. For probabilistic forecast, thresholds must be specified using "ge" convention.

To indicate that a forecast field should be treated as probabilities, set the **prob** field to TRUE. Probability fields should contain values in the range [0, 1] or [0, 100]. However, when MET encounters a probability field with a range [0, 100], it will automatically rescale it to be [0, 1] before applying the probabilistic verification methods.

The **obs** group can be defined in the same way as the forecast group. However, for most users, setting obs=fcst to use the same settings will be most valuable. The variables are identical those in the **fcst** except that they apply to the observation field.

For each observation field, one or more thresholds can be specified for use in computing discrete statistics. Specification is done exactly as for **cat_thresh**.

When verifying winds via either the u and v components or the speed and direction, it can be desirable to eliminate winds below a certain speed. This threshold filters the winds based on speed, even when u and v winds are input. Format is the same as for **cat_thresh**.

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.

```
beg = -5400;
end =  5400;
```

Each gridded forecast file has a valid time associated with it. The **beg** and **end** variables define a time window in seconds around the valid time of the forecast file for the observations to be matched to it. For a forecast valid time, v, observations with a valid time falling in the time window [v+beg, v+end] will be used.

```
mask = {
   grid = [ "FULL" ];
   poly = [];
   sid  = "";
};
```

The **mask** group contains variables to define the verification grid. The **grid** 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.

The **poly** variable contains a comma-separated list of files that define verification masking regions. These masking regions may be specified in two ways: as a lat/lon polygon or using a gridded data file such as the NetCDF output of the Gen-Poly-Mask tool.

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.

Alternatively, any gridded data file that MET can read may be used to define a verification masking region. Users must specify a description of the field to be used from the input file and, optionally, may specify a threshold to be applied to that field. Any grid point where the resulting field is 0, the mask is turned off. Any grid point where it is non-zero, the mask is turned on.

The **sid** variable contains a filename that 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 = PCTILE;
   rep_prop = 1.0;
   n_rep    = 1000;
   rng      = "mt19937";
   seed     = "";
};
```

The **boot** group defines the parameters to be used in calculation of bootstrap confidence intervals. The **interval** variable indicates what method should be used for computing bootstrap confidence intervals. A value of BCA indicates that the highly accurate but computationally intensive BCa (bias-corrected percentile) method should be used. A value of PCTILE indicates that the somewhat less accurate but efficient percentile method should be used.

The **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.

The **n_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 that 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.

The **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.

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

value here ensures that the bootstrap confidence intervals will be computed the same over multiple runs of the same data.

```
interp = {
   vld_thresh = 1.0;

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

The **interp** group specifies the parameters for grid interpolation or smoothing.

The **vld_thresh** 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.

The **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, LS_FIT, and BILIN for the minimum, maximum, median, unweighted mean, distance-weighted mean, a least squares fit, and bilinear interpolation. 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.

The **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 **width** value specifies the width of that square. A **width** value of 1 is interpreted as the nearest neighbor model grid point to the observation point. A **width** of 2 defines a 2 x 2 square of grid points around the observation point (the 4 closest model grid points), while a **width** 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.

```
output_flag = {
   fho    = BOTH;
   ctc    = BOTH;
```

```
   cts     = BOTH;
   mctc    = BOTH;
   mcts    = BOTH;
   cnt     = BOTH;
   sl1l2   = BOTH;
   sal1l2  = BOTH;
   vl1l2   = BOTH;
   val1l2  = BOTH;
   pct     = BOTH;
   pstd    = BOTH;
   pjc     = BOTH;
   prc     = BOTH;
   mpr     = BOTH;
};
```

The **output_flag** array controls the type of output that the Point-Stat tool generates. Each flag corresponds to an output line type in the STAT file. Setting the flag to NONE indicates that the line type should not be generated. Setting the flag to STAT indicates that the line type should be written to the STAT file only. Setting the flag to BOTH indicates that the line type should be written to the STAT file as well as a separate ASCII file where the data is grouped by line type. The output flags correspond to the following output line types:

1. FHO for Forecast, Hit, Observation Rates
2. CTC for Contingency Table Counts
3. CTS for Contingency Table Statistics
4. MCTC for Multi-category Contingency Table Counts
5. MCTS for Multi-category Contingency Table Statistics
6. CNT for Continuous Statistics
7. SL1L2 for Scalar L1L2 Partial Sums
8. SAL1L2 for Scalar Anomaly L1L2 Partial Sums when climatological data is supplied
9. VL1L2 for Vector L1L2 Partial Sums
10. VAL1L2 for Vector Anomaly L1L2 Partial Sums when climatological data is supplied
11. PCT for Contingency Table counts for Probabilistic forecasts
12. PSTD for contingency table Statistics for Probabilistic forecasts with Dichotomous outcomes
13. PJC for Joint and Conditional factorization for Probabilistic forecasts
14. PRC for Receiver Operating Characteristic for Probabilistic forecasts
15. MPR for Matched Pair data

Note that the first two line types are easily derived from each other. Users are free to choose which measures are most desired. All of the line types are described in more detail in Section 4.3.3.

Note that generating matched pair data (MPR lines) for a large number of cases is generally not recommended. The MPR lines create very large output files and are only intended for use on a small set of cases.

---

rank_corr_flag = TRUE;

The **rank_corr_flag** variable may be set to TRUE or FALSE 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 FALSE should improve the runtime performance.

---

duplicate_flag = NONE;

The reporting mechanism for this feature can be activated by specifying a verbosity level of three or higher. The report will show information about where duplicates were detected and which observations were used in those cases.

---

tmp_dir = "/tmp";

This option specifies the directory where temp files should be written by the Point-Stat tool.

---

output_prefix = "";

This option specifies a string to be used in the output file name. It can be useful for keeping results for different models or variables from overwriting each other.

---

version = "V4.0";

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 STAT and, optionally, ASCII format. The ASCII output duplicates the STAT output but has the data organized by line type. The output files will

be written to the default output directory or the directory specified using the "-outdir" command line option. The output STAT file will be named using the following naming convention: **`point_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV.stat`** where **PREFIX** indicates the user-defined output prefix, **YYYMMDDHH indicates the forecast lead time and YYYYMMDD_HHMMSS** indicates the forecast valid time. The output ASCII files are named similarly: **`point_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV_TYPE.txt`** where **TYPE** is one of **`fho`**, **`ctc`**, **`cts`**, **`cnt, mctc, mcts, pct, pstd, pjc, prc`**, **`sl1l2`**, **`sal1l2`**, **`vl1l2`**, or **`val1l2`** to indicate the line type it contains.

The first set of header columns are common to all of the output files generated by the Point-Stat tool. Tables describing the contents of the header columns and the contents of the additional columns for each line type are listed in the following tables.

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

HEADER		
Column Number	**Header Column Name**	**Description**
1	VERSION	Version number (set to 3.0)
2	MODEL	User provided text string designating model name
3	FCST_LEAD	Forecast lead time in HHMMSS format
4	FCST_VALID_BEG	Forecast valid start time in YYYYMMDDHH format
5	FCST_VALID_END	Forecast valid end time in YYYYMMDDHH format
6	OBS_LEAD	Observation lead time in HHMMSS format
7	OBS_VALID_BEG	Observation valid start time in YYYYMMDDHH format
8	OBS_VALID_END	Observation valid end time in YYYYMMDDHH format
9	FCST_VAR	Model variable
10	FCST_LEV	Selected Vertical level for forecast
11	OBS_VAR	Observed variable
12	OBS_LEV	Selected Vertical level for observations
13	OBTYPE	Type of observation selected
14	VX_MASK	Verifying masking region indicating the masking grid or polyline region applied
15	INTERP_MTHD	Interpolation method applied to forecasts
16	INTERP_PNTS	Number of points used in interpolation method
17	FCST_THRESH	The threshold applied to the forecast
18	OBS_THRESH	The threshold applied to the observations
19	COV_THRESH	NA in Point-Stat
20	ALPHA	Error percent value used in confidence intervals
21	LINE_TYPE	

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

FHO OUTPUT FORMAT		
Column Number	**FHO Column Name**	**Description**
21	FHO	Forecast, Hit, Observation line type
22	TOTAL	Total number of matched pairs
23	F_RATE	Forecast rate
24	H_RATE	Hit rate
25	O_RATE	Observation rate

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

CTC OUTPUT FORMAT		
Column Number	**CTC Column Name**	**Description**
21	CTC	Contingency Table Counts line type
22	TOTAL	Total number of matched pairs
23	FY_OY	Number of forecast yes and observation yes
24	FY_ON	Number of forecast yes and observation no
25	FN_OY	Number of forecast no and observation yes
26	FN_ON	Number of forecast no and observation no

Table 4-5. *Format information for CTS (Contingency Table Statistics) output line type.*

CTS OUTPUT FORMAT		
Column Number	**CTS Column Name**	**Description**
21	CTS	Contingency Table Statistics line type
22	TOTAL	Total number of matched pairs
23-27	BASER, BASER_NCL, BASER_NCU, BASER_BCL, BASER_BCU	Base rate including normal and bootstrap upper and lower confidence limits
28-32	FMEAN, FMEAN_NCL, FMEAN_NCU, FMEAN_BCL, FMEAN_BCU,	Forecast mean including normal and bootstrap upper and lower confidence limits
33-37	ACC, ACC_NCL, ACC_NCU, ACC_BCL,	Accuracy including normal and bootstrap upper and lower confidence limits

CTS OUTPUT FORMAT		
Column Number	**CTS Column Name**	**Description**
	ACC_BCU	
38-40	FBIAS, FBIAS_BCL, FBIAS_BCU	Frequency Bias including bootstrap upper and lower confidence limits
41-45	PODY, PODY_NCL, PODY_NCU, PODY_BCL, PODY_BCU	Probability of detecting yes including normal and bootstrap upper and lower confidence limits
46-50	PODN, PODN_NCL, PODN_NCU, PODN_BCL, PODN_BCU	Probability of detecting no including normal and bootstrap upper and lower confidence limits
51-55	POFD, POFD_NCL, POFD_NCU, POFD_BCL, POFD_BCU	Probability of false detection including normal and bootstrap upper and lower confidence limits
56-60	FAR, FAR_NCL, FAR_NCU, FAR_BCL, FAR_BCU	False alarm ratio including normal and bootstrap upper and lower confidence limits
61-65	CSI, CSI_NCL, CSI_NCU, CSI_BCL, CSI_BCU	Critical Success Index including normal and bootstrap upper and lower confidence limits
66-68	GSS, GSS_BCL, GSS_BCU	Gilbert Skill Score including bootstrap upper and lower confidence limits
69-73	HK, HK_NCL, HK_NCU, HK_BCL, HK_BCU	Hanssen-Kuipers Discriminant including normal and bootstrap upper and lower confidence limits
74-76	HSS, HSS_BCL, HSS_BCU	Heidke Skill Score including bootstrap upper and lower confidence limits
77-81	ODDS, ODDS_NCL, ODDS_NCU,	Odds Ratio including normal and bootstrap upper and lower confidence limits

CTS OUTPUT FORMAT		
Column Number	**CTS Column Name**	**Description**
	ODDS_BCL, ODDS_BCU	

Table 4-6. *Format information for CNT(Continuous Statistics) output line type.*

CNT OUTPUT FORMAT		
Column Number	**CNT Column Name**	**Description**
21	CNT	Continuous statistics line type
22	TOTAL	Total number of matched pairs
23-27	FBAR, FBAR_NCL, FBAR_NCU, FBAR_BCL, FBAR_BCU	Forecast mean including normal and bootstrap upper and lower confidence limits
28-32	FSTDEV, FSTDEV_NCL, FSTDEV_NCU, FSTDEV_BCL, FSTDEV_BCU	Standard deviation of the forecasts including normal and bootstrap upper and lower confidence limits
33-37	OBAR, OBAR_NCL, OBAR_NCU, OBAR_BCL, OBAR_BCU	Observation mean including normal and bootstrap upper and lower confidence limits
38-42	OSTDEV, OSTDEV_NCL, OSTDEV_NCU, OSTDEV_BCL, OSTDEV_BCU	Standard deviation of the observations including normal and bootstrap upper and lower confidence limits
43-47	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
48	SP_CORR	Spearman's rank correlation coefficient
49	KT_CORR	Kendall's tau statistic
50	RANKS	Number of ranks used in computing Kendall's tau statistic
51	FRANK_TIES	Number of tied forecast ranks used in computing Kendall's tau statistic
52	ORANK_TIES	Number of tied observation ranks used in computing Kendall's tau statistic

CNT OUTPUT FORMAT		
Column Number	**CNT Column Name**	**Description**
53-57	ME, ME_NCL, ME_NCU, ME_BCL, ME_BCU	Mean error (F-O) including normal and bootstrap upper and lower confidence limits
58-62	ESTDEV, ESTDEV_NCL, ESTDEV_NCU, ESTDEV_BCL, ESTDEV_BCU	Standard deviation of the error including normal and bootstrap upper and lower confidence limits
63-65	MBIAS, MBIAS_BCL, MBIAS_BCU	Multiplicative bias including bootstrap upper and lower confidence limits
66-68	MAE, MAE_BCL, MAE_BCU	Mean absolute error including bootstrap upper and lower confidence limits
68-71	MSE, MSE_BCL, MSE_BCU	Mean squared error including bootstrap upper and lower confidence limits
72-74	BCMSE, BCMSE_BCL, BCMSE_BCU	Bias-corrected mean squared error including bootstrap upper and lower confidence limits
75-77	RMSE, RMSE_BCL, RMSE_BCU	Root mean squared error including bootstrap upper and lower confidence limits
78-92	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	10^{th}, 25^{th}, 50^{th}, 75^{th}, and 90^{th} percentiles of the error including bootstrap upper and lower confidence limits

Table 4-7. *Format information for MCTC (Multi-category Contingency Table Count) output line type.*

MCTC OUTPUT FORMAT		
Column Number	**CTC Column Name**	**Description**
21	MCTC	Multi-category Contingency Table Counts line type
22	TOTAL	Total number of matched pairs
23	DIM	Dimension of the contingency table.
24-??	Fi_Oj	Count of events in forecast category i and observation category j, with the observations incrementing first.

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

MCTS OUTPUT FORMAT		
Column Number	**CTS Column Name**	**Description**
21	MCTS	Multi-category Contingency Table Statistics line type
22	TOTAL	Total number of matched pairs
23	N_CAT	The total number of categories in each of dimension of the contingency table. So the total number of cells is N_CAT*N_CAT.
24-28	ACC, ACC_NCL, ACC_NCU, ACC_BCL, ACC_BCU	Accuracy, normal confidence limits and bootstrap confidence limits
29-31	HK, HK_BCL, HK_BCU	Hanssen and Kuipers Discriminant and bootstrap confidence limits
32-34	HSS, HSS_BCL, HSS_BCU	Heidke Skill Score and bootstrap confidence limits
35-37	GER, GER_BCL, GER_BCU	Gerrity Score and bootstrap confidence limits

Table 4-9. *Format information for PCT (Contingency Table Counts for Probabilistic forecasts) output line type.*

PCT OUTPUT FORMAT		
Column Number	**PCT Column Name**	**Description**
21	PCT	Probability contingency table count line type
22	TOTAL	Total number of matched pairs

PCT OUTPUT FORMAT		
Column Number	**PCT Column Name**	**Description**
23	N_THRESH	Number of probability thresholds
24	THRESH_i	The i^{th} probability threshold value (repeated)
25	OY_i	Number of observation yes when forecast is between the i^{th} and $i+1^{th}$ probability thresholds (repeated)
26	ON_i	Number of observation no when forecast is between the i^{th} and $i+1^{th}$ probability thresholds (repeated)
*	THRESH_n	Last probability threshold value

Table 4-10. *Format information for PSTD (Contingency Table Statistics for Probabilistic forecasts) output line type.*

PSTD OUTPUT FORMAT		
Column Number	**PSTD Column Name**	**Description**
21	PSTD	Probabilistic statistics for dichotomous outcome line type
22	TOTAL	Total number of matched pairs
23	N_THRESH	Number of probability thresholds
24-26	BASER, BASER_NCL, BASER_NCU	The Base Rate, including normal upper and lower confidence limits
27	RELIABILITY	Reliability
28	RESOLUTION	Resolution
29	UNCERTAINTY	Uncertainty
30	ROC_AUC	Area under the receiver operating characteristic curve
312-33	BRIER, BRIER_NCL, BRIER_NCU	Brier Score including normal upper and lower confidence limits
34	THRESH_i	The i^{th} probability threshold value (repeated)

Table 4-11. *Format information for PJC (Joint and Conditional factorization for Probabilistic forecasts) output line type.*

PJC OUTPUT FORMAT		
Column Number	**PJC Column Name**	**Description**
21	PJC	Probabilistic Joint/Continuous line type
22	TOTAL	Total number of matched pairs
23	N_THRESH	Number of probability thresholds
24	THRESH_i	The i^{th} probability threshold value (repeated)

PJC OUTPUT FORMAT		
Column Number	**PJC Column Name**	**Description**
25	OY_TP_i	Number of observation yes when forecast is between the i^{th} and $i+1^{th}$ probability thresholds as a proportion of the total OY (repeated)
26	ON_TP_i	Number of observation no when forecast is between the i^{th} and $i+1^{th}$ probability thresholds as a proportion of the total ON (repeated)
27	CALIBRATION _i	Calibration when forecast is between the i^{th} and $i+1^{th}$ probability thresholds (repeated)
28	REFINEMENT_i	Refinement when forecast is between the i^{th} and $i+1^{th}$ probability thresholds (repeated)
29	LIKELIHOOD_i	Likelihood when forecast is between the i^{th} and $i+1^{th}$ probability thresholds (repeated)
30	BASER_i	Base rate when forecast is between the i^{th} and $i+1^{th}$ probability thresholds (repeated)
*	THRESH_n	Last probability threshold value

Table 4-12. *Format information for PRC (PRC for Receiver Operating Characteristic for Probabilistic forecasts) output line type.*

PRC OUTPUT FORMAT		
Column Number	**PRC Column Name**	**Description**
21	PRC	Probability ROC points line type
22	TOTAL	Total number of matched pairs
23	N_THRESH	Number of probability thresholds
24	THRESH_i	The i^{th} probability threshold value (repeated)
25	PODY_i	Probability of detecting yes when forecast is between the i^{th} and $i+1^{th}$ probability thresholds (repeated)
26	POFD_i	Probability of false detection when forecast is between the i^{th} and $i+1^{th}$ probability thresholds (repeated)
*	THRESH_n	Last probability threshold value

Table 4-13. *Format information for SL1L2 (Scalar Partial Sums) output line type.*

SL1L2 OUTPUT FORMAT		
Column Number	**SL1L2 Column Name**	**Description**
21	SL1L2	Scalar L1L2 line type
22	TOTAL	Total number of matched pairs of forecast (f) and observation (o)
23	FBAR	Mean(f)

SL1L2 OUTPUT FORMAT		
Column Number	**SL1L2 Column Name**	**Description**
24	OBAR	Mean(o)
25	FOBAR	Mean(f*o)
26	FFBAR	Mean(f^2)
27	OOBAR	Mean(o^2)

Table 4-14. *Format information for SAL1L2 (Scalar Anomaly Partial Sums) output line type.*

SAL1L2 OUTPUT FORMAT		
Column Number	**SAL1L2 Column Name**	**Description**
21	SAL1L2	Scalar Anomaly L1L2 line type
22	TOTAL	Total number of matched triplets of forecast (f), observation (o), and climatological value (c)
23	FABAR	Mean(f-c)
24	OABAR	Mean(o-c)
25	FOABAR	Mean((f-c)*(o-c))
26	FFABAR	Mean($(f-c)^2$)
27	OOABAR	Mean($(o-c)^2$)

Table 4-15. *Format information for VL1L2 (Vector Partial Sums) output line type.*

VL1L2 OUTPUT FORMAT		
Column Number	**VL1L2 Column Name**	**Description**
21	VL1L2	Vector L1L2 line type
22	TOTAL	Total number of matched pairs of forecast winds (uf, vf) and observation winds (uo, vo)
23	UFBAR	Mean(uf)
24	VFBAR	Mean(vf)
25	UOBAR	Mean(uo)
26	VOBAR	Mean(vo)
27	UVFOBAR	Mean(uf*uo+vf*vo)
28	UVFFBAR	Mean(uf^2+vf^2)
29	UVOOBAR	Mean(uo^2+vo^2)

Table 4-18. *Format information for VAL1L2 (Vector Anomaly Partial Sums) output line type.*

VAL1L2 OUTPUT FILE		
Column Number	**VAL1L2 Column Name**	**Description**
21	VAL1L2	Vector Anomaly L1L2 line type
22	TOTAL	Total number of matched triplets of forecast winds (uf, vf), observation winds (uo, vo), and climatological winds (uc, vc)
23	UFABAR	Mean(uf-uc)
24	VFABAR	Mean(vf-vc)
25	UOABAR	Mean(uo-uc)
26	VOABAR	Mean(vo-vc)
27	UVFOABAR	Mean((uf-uc)*(uo-uc)+(vf-vc)*(vo-vc))
28	UVFFABAR	Mean$((uf-uc)^2+(vf-vc)^2)$
29	UVOOABAR	Mean$((uo-uc)^2+(vo-vc)^2)$

Table 4-19. *Format information for MPR (Matched Pair) output line type.*

MPR OUTPUT FORMAT		
Column Number	**MPR Column Name**	**Description**
21	MPR	Matched Pair line type
22	TOTAL	Total number of matched pairs
23	INDEX	Index for the current matched pair
24	OBS_SID	Station Identifier of observation
25	OBS_LAT	Latitude of the observation in degrees north
26	OBS_LON	Longitude of the observation in degrees east
27	OBS_LVL	Pressure level of the observation in hPa or accumulation interval in hours
28	OBS_ELV	Elevation of the observation in meters above sea level
29	FCST	Forecast value interpolated to the observation location
30	OBS	Observation value
31	CLIMO	Climatological value

The STAT output files described for `point_stat` may be used as inputs to the STAT Analysis tool. For more information on using the STAT Analysis tool to create stratifications and aggregations of the STAT files produced by `point_stat`, please see Chapter 8.

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 grid points in the region of interest are matched to observation grid points on the same grid. All the matched grid points are used to compute the verification statistics. The Grid-Stat tool functions in much the same way as the Point-Stat tool, except that no interpolation is required because the forecasts and observations are on the same grid. However, the interpolation parameters may be used to perform a smoothing operation on the forecast and observation fields prior to verification. In addition to traditional verification approaches, the Grid-Stat tool includes 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 three types of statistics: measures for *categorical* variables, measures for *continuous* variables, and measures for probabilistic forecasts. These categories of measures are briefly described here; specific descriptions of all measures are provided in Appendix C. Additional information can be found in Wilks (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 Grid-Stat tool also computes partial sums and other FHO statistics that are produced by the NCEP verification system. These statistics are also described in Appendix C.

Measures for categorical variables

Categorical verification statistics are used to evaluate forecasts that are in the form of a discrete set of categories rather than on a continuous scale. Currently, Grid-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.

Measures for probabilistic forecasts and dichotomous outcomes
For probabilistic forecasts, many verification measures are based on reliability, accuracy and bias. However, it also is of interest to investigate joint and conditional distributions of the forecasts and the observations, as in Wilks (2006). See Appendix C for specific information.

Probabilistic forecast values are assumed to have a range of either 0 to 1 or 0 to 100. If the max data value is > 1, we assume the data range is 0 to 100, and divide all the values by 100. If the max data value is <= 1, then we use the values as is. Further, thresholds are applied to the probabilities with equality on the lower end. For example, with a forecast probability *p*, and thresholds *t1* and t*2*, the range is defined as: *t1* <= *p* < *t2*. The exception is for the highest set of thresholds, when the range includes 1: *t1* <= *p* <= 1.

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 Statistical confidence intervals

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

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]
        [-log file]
        [-v level]
```

`grid_stat` has three required arguments and up to six 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 **`-log` file** option directs output and errors to the specified log file. All messages will be written to that file as well as cout and cerr. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no logfile.

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

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. **A web tool that generates config file text is available via a link from the MET Users' web site.** 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 tool.

```
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 STAT output so that verification statistics from multiple models may be differentiated. The value listed above is simply set to "WRF".

```
fcst = {
   wind_thresh = [ NA ];

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

};
obs = fcst;
```

The **fcst** group contains a list of model variables, corresponding vertical levels, and thresholds to be verified. Each field is specified as a GRIB code or abbreviation followed by an accumulation or vertical level indicator for GRIB files or as a variable name followed by a list of dimensions for NetCDF files output from p_interp or MET.

For GRIB files, 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**", "**PNNN-NNN**", "**LNNN**", or "**RNNN**" must follow each GRIB code. These indicate an accumulation interval, a single vertical level, a single pressure level, a range of pressure levels, a generic level, and a specific GRIB record number, respectively. "**NNN**" indicates the accumulation or level value.

To specify verification fields for NetCDF files, use var_name(i,...,j,*,*) for a single field. Here, var_name is the name of the NetCDF variable, i,...,j specifies fixed dimension values, and *,* specifies the two dimensions for the gridded field.

The values listed above indicate that specific humidity, temperature, height, wind speed, 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. Wind speed is typically not available in model files, but if it is, it will be used directly. However, if wind speed is unavailable but the U and V components of the wind are included, the wind speed will be calculated automatically by MET, provided that WIND is included in the **field**.

To indicate that a forecast field should be treated as probabilities, include the prob=TRUE indicator. Probability fields should contain values in the range [0, 1] or [0, 100]. However, when MET encounters a probability field with a range [0, 100], it will automatically rescale it to be [0, 1] before applying the probabilistic verification methods.

For each **cat_thresh** 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. Multiple thresholds may be applied to each field by providing 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. For probabilistic forecast, thresholds must be specified using "ge" convention.

When verifying winds via either the u and v components or the speed and direction, it can be desirable to eliminate winds below a certain speed. This threshold filters the winds based on speed, even when u and v winds are input. Format is the same as for **cat_thresh**.

```
mask = {
   grid = [ "FULL" ];
   poly = [];
};
```

The **grid** variable in the **mask** group 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.

The **poly** variable contains a comma-separated list of files that define verification masking regions. These masking regions may be specified in two ways: as a lat/lon polygon or using a gridded data file such as the NetCDF output of the Gen-Poly-Mask tool.

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.

Alternatively, any gridded data file that MET can read may be used to define a verification masking region. Users must specify a description of the field to be used from the input file and, optionally, may specify a threshold to be applied to that field.

Any grid point where the resulting field is 0, the mask is turned off. Any grid point where it is non-zero, the mask is turned on.

```
ci_alpha  = [ 0.05 ];

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 0;
   rng      = "mt19937";
   seed     = "";
};
```

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.

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

The **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.

The **n_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 that 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.

The **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.

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

```
interp = {
   field      = BOTH;
   vld_thresh = 1.0;

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

The **interp** group contains a list of operations to be performed on the forecast or observation field prior to performing verification. The valid **method** values that may be listed are **MIN**, **MAX**, **MEDIAN**, and **UW_MEAN** for the minimum, maximum, median, and unweighted mean. If multiple interpolation methods are provided, then statistics will be computed separately for each type of smoothing operation. These methods are described in Section 4.2.1.

The **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, and thus must be odd. The **width** value specifies the width of that square.

The **field** controls how the interpolation should be applied. A value of **FCST** will smooth, e.g. apply the interpolation, to only the forecast field. A value of **OBS** applies interpolation to only the observation field. To smooth both the forecast and observation fields, set the interp_flag to **BOTH**.

The **vld_thresh** 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 computed. 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.

```
nbrhd = {
   vld_thresh = 1.0;
   width      = [ 1 ];
   cov_thresh = [ >=0.5 ];
}
```

The **nbrhd** group contains a list of values to be used in defining the neighborhood to be used when computing neighborhood verification statistics. The neighborhood is simply a square centered on the current point and the **width** value specifies the width of that square as an odd integer.

The **vld_thresh** 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.

The **cov_thresh** variable contains a comma separated list of thresholds to be applied to the neighborhood coverage field. The coverage is the proportion of forecast points in the neighborhood that exceed the forecast threshold. For example, if 10 of the 25 forecast grid points contain values larger than a threshold of 2, then the coverage is 10/25 = 0.4. If the coverage threshold is set to 0.5, then this neighborhood is considered to be a “No” forecast.

```
output_flag = {
   fho    = BOTH;
   ctc    = BOTH;
   cts    = BOTH;
   mctc   = BOTH;
   mcts   = BOTH;
   cnt    = BOTH;
   sl1l2  = BOTH;
   vl1l2  = BOTH;
   pct    = BOTH;
   pstd   = BOTH;
   pjc    = BOTH;
   prc    = BOTH;
   nbrctc = BOTH;
   nbrcts = BOTH;
   nbrcnt = BOTH;
};
```

The **output_flag** array controls the type of output that the Grid-Stat tool generates. Each flag corresponds to an output line type in the STAT file except for the last one. Setting the flag to **NONE** indicates that the line type should not be generated. Setting the flag to **STAT** indicates that the line type should be written to the STAT file only. Setting the flag to **BOTH** indicates that the line type should be written to the STAT file as well as a separate ASCII file where the data are grouped by line type. The first fifteen output flags correspond to the following types of output line types:

1. FHO for Forecast, Hit, Observation Rates
2. CTC for Contingency Table Counts
3. CTS for Contingency Table Statistics
4. MCTC for Multi-Category Contingency Table Counts
5. MCTS for Multi-Category Contingency Table Statistics
6. CNT for Continuous Statistics
7. SL1L2 for Scalar L1L2 Partial Sums
8. VL1L2 for Vector L1L2 Partial Sums
9. PCT for Contingency Table Counts for Probabilistic forecasts
10. PSTD for Contingency Table Statistics for Probabilistic forecasts
11. PJC for Joint and Conditional factorization for Probabilistic forecasts
12. PRC for Receiver Operating Characteristic for Probabilistic forecasts
13. NBRCTC for Neighborhood Contingency Table Counts
14. NBRCTS for Neighborhood Contingency Table Statistics
15. NBRCNT for Neighborhood Continuous Statistics

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

```
nc_pairs_flag = TRUE;
```

This flag indicates whether or not the matched pair and forecast minus observation difference fields should be written to a NetCDF file. Setting the flag to **TRUE** indicates that the NetCDF file should be created, while setting it to **FALSE** disables its creation.

```
rank_corr_flag = TRUE;
```

The **rank_corr_flag** variable may be set to **TRUE** or **FALSE** 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 **FALSE** should improve the runtime performance.

tmp_dir = "/tmp";

This parameter indicates the directory where Grid Stat should write temporary files.

---

output_prefix = "";

This option specifies a string to be used in the output file name. It can be useful for keeping results for different models or variables from overwriting each other.

---

version = "V4.0";

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 STAT and, optionally, ASCII and NetCDF formats. The ASCII output duplicates the STAT output but has the data organized by line type. The output files are written to the default output directory or the directory specified by the **-outdir** command-line option.

The output STAT file is named using the following naming convention: **grid_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV.stat** where **PREFIX** indicates the user-defined output prefix, **YYYMMDDHH** indicates the forecast lead time and **YYYYMMDD_HHMMSS** indicates the forecast valid time. The output ASCII files are named similarly: **point_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV_TYPE.txt** where **TYPE** is one of **fho**, **ctc**, **cts**, **cnt**, **sl1l2, vl1l2**, **pct**, **pstd**, **pjc**, **prc**, **nbrctc**, **nbrcts**, and **nbrcnt** to indicate the line type it contains.

The format of the STAT and ASCII output of the Grid-Stat tool are the same as the format of the STAT and ASCII output of the Point-Stat tool with the exception of the 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 STAT 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	VERSION	Version number (set to 3.0)
2	MODEL	User provided text string designating model name
3	FCST_LEAD	Forecast lead time in HHMMSS format
4	FCST_VALID_BEG	Forecast valid start time in YYYYMMDDHH format
5	FCST_VALID_END	Forecast valid end time in YYYYMMDDHH format
6	OBS_LEAD	Observation lead time in HHMMSS format
7	OBS_VALID_BEG	Observation valid start time in YYYYMMDDHH format
8	OBS_VALID_END	Observation valid end time in YYYYMMDDHH format
9	FCST_VAR	Model variable
10	FCST_LEV	Selected Vertical level for forecast
11	OBS_VAR	Observed variable
12	OBS_LEV	Selected Vertical level for observations
13	OBTYPE	Type of observation selected
14	VX_MASK	Verifying masking region indicating the masking grid or polyline region applied
15	INTERP_MTHD	Interpolation method applied to forecast field
16	INTERP_PNTS	Number of points used by interpolation method
17	FCST_THRESH	The threshold applied to the forecast
18	OBS_THRESH	The threshold applied to the observations
19	COV_THRESH	Coverage threshold for neighborhood methods
20	ALPHA	Error percent value used in confidence intervals
21	LINE_TYPE	

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

NBRCTC OUTPUT FORMAT		
Column Number	**NBRCTC Column Name**	**Description**
21	NBRCTC	Neighborhood Contingency Table Counts line type
22	TOTAL	Total number of matched pairs
23	FY_OY	Number of forecast yes and observation yes
24	FY_ON	Number of forecast yes and observation no
25	FN_OY	Number of forecast no and observation yes
26	FN_ON	Number of forecast no and observation no

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

NBRCTS OUTPUT FORMAT		
Column Number	**NBRCTS Column Name**	**Description**
21	NBRCTS	Neighborhood Contingency Table Statistics line type
22	TOTAL	Total number of matched pairs
23-27	BASER, BASER_NCL, BASER_NCU, BASER_BCL, BASER_BCU	Base rate including normal and bootstrap upper and lower confidence limits
28-32	FMEAN, FMEAN_NCL, FMEAN_NCU, FMEAN_BCL, FMEAN_BCU,	Forecast mean including normal and bootstrap upper and lower confidence limits
33-37	ACC, ACC_NCL, ACC_NCU, ACC_BCL, ACC_BCU	Accuracy including normal and bootstrap upper and lower confidence limits
38-40	FBIAS, FBIAS_BCL, FBIAS_BCU	Frequency Bias including bootstrap upper and lower confidence limits
41-45	PODY, PODY_NCL, PODY_NCU, PODY_BCL, PODY_BCU	Probability of detecting yes including normal and bootstrap upper and lower confidence limits
46-50	PODN, PODN_NCL, PODN_NCU, PODN_BCL, PODN_BCU	Probability of detecting no including normal and bootstrap upper and lower confidence limits
51-55	POFD, POFD_NCL, POFD_NCU, POFD_BCL, POFD_BCU	Probability of false detection including normal and bootstrap upper and lower confidence limits
56-60	FAR, FAR_NCL, FAR_NCU, FAR_BCL, FAR_BCU	False alarm ratio including normal and bootstrap upper and lower confidence limits
61-65	CSI,	Critical Success Index including normal and bootstrap

NBRCTS OUTPUT FORMAT		
Column Number	**NBRCTS Column Name**	**Description**
	CSI_NCL, CSI_NCU, CSI_BCL, CSI_BCU	upper and lower confidence limits
66-68	GSS, GSS_BCL, GSS_BCU	Gilbert Skill Score including bootstrap upper and lower confidence limits
69-73	HK, HK_NCL, HK_NCU, HK_BCL, HK_BCU	Hanssen-Kuipers Discriminant including normal and bootstrap upper and lower confidence limits
74-76	HSS, HSS_BCL, HSS_BCU	Heidke Skill Score including bootstrap upper and lower confidence limits
77-81	ODDS, ODDS_NCL, ODDS_NCU, ODDS_BCL, ODDS_BCU	Odds Ratio including normal and bootstrap upper and lower confidence limits

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

NBRCNT OUTPUT FORMAT		
Column Number	**NBRCNT Column Name**	**Description**
21	NBRCNT	Neighborhood Continuous statistics line type
22	TOTAL	Total number of matched pairs
23-25	FBS, FBS_BCL, FBS_BCU	Fractions Brier Score including bootstrap upper and lower confidence limits
26-28	FSS, FSS_BCL, FSS_BCU	Fractions Skill Score including bootstrap upper and lower confidence limits

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_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV_pairs.nc`**. Commonly available NetCDF utilities such as `ncdump` or `ncview` may be used to view the contents of the output file.

The output NetCDF file contains the dimensions and variables shown in the following Tables 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**
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 STAT output files described for `grid_stat` may be used as inputs to the STAT Analysis tool. For more information on using the STAT Analysis tool to create stratifications and aggregations of the STAT files produced by `grid_stat`, please see Chapter 8.

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/field2 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:

$$C(x,y) = \sum \phi(u,v) f(x-u)(y-v) .$$

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*:

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

The parameters *R* and *H* are not independent. They are related by the requirement that the integral of 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*:

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

The objects are the connected regions where *M* = 1. Finally, the raw data are restored to object interiors to obtain the object field, *F*:

$$F(x,y) = M(x,y) f(x,y).$$

Thus, 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.

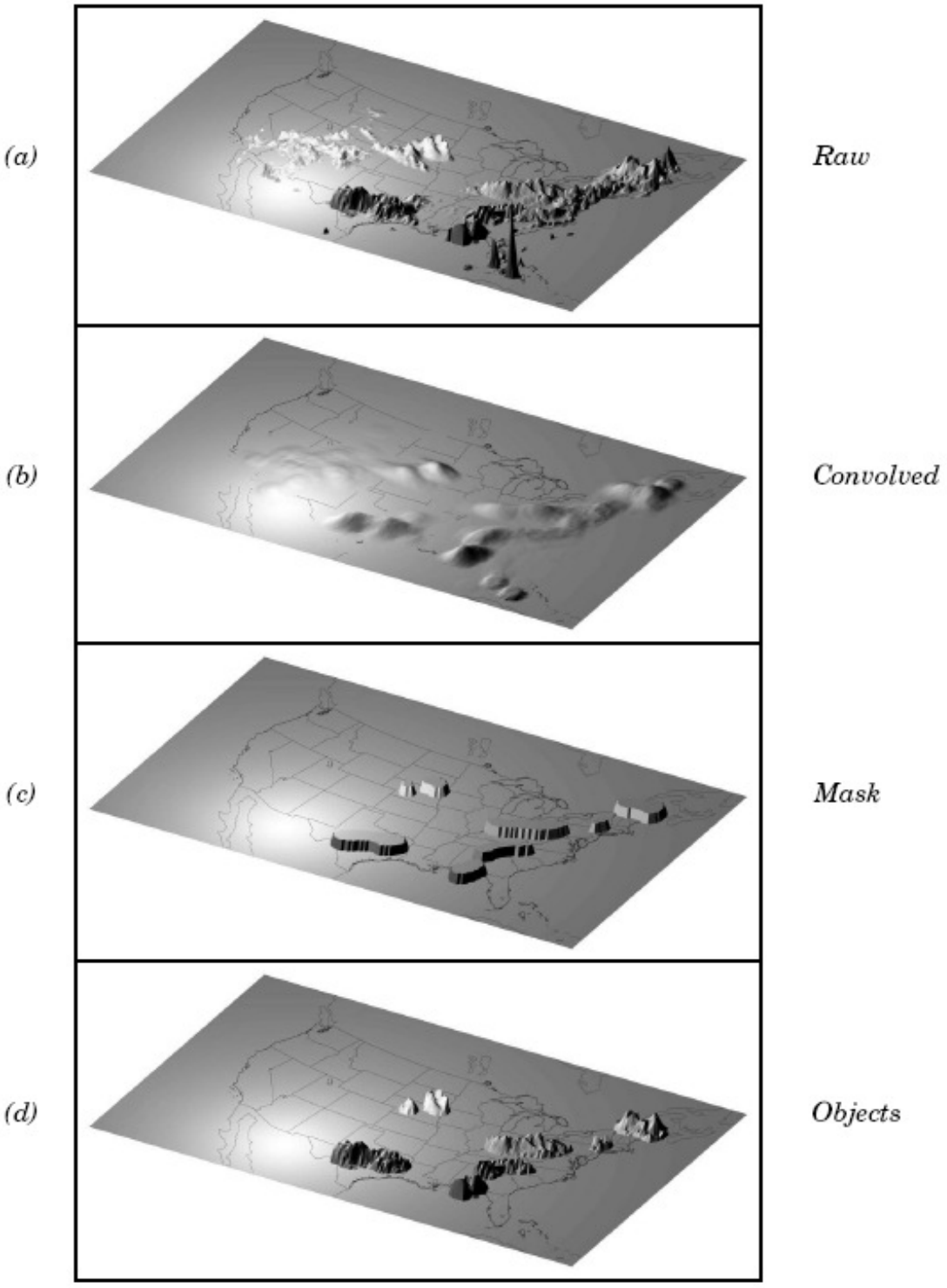


Figure 6-1*: Example of an application of the MODE object identification process to a model precipitation field.*

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 cluster objects and their attributes. Total interest for object pairs is also output, as are percentiles of intensity inside the objects. The output file is in a simple flat ASCII tabular format (with one header line) and thus should be easily readable by just about any programming language, scripting language, or statistics package. (See the examples using `awk` in Chapter 10.) Refer to Section 6.3.3 for lists of the statistics included in the `mode` output files. Example scripts will be posted on the MET website in the future.

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]
        [-log file]
        [-v level]
```

The MODE tool has three required arguments and can accept several optional arguments.

Required arguments for **`mode`**

1. The **fcst_file** argument indicates the 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 **`-log` file** option directs output and errors to the specified log file. All messages will be written to that file as well as cout and cerr. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no logfile.

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

An example of the 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. **A web tool that generates config file text is available via a link from the MET Users' web site.** 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 tool.

```
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 = {
   field = {
      name  = "APCP";
      level = "A03";
   };

   raw_thresh        = >=0.0;
   conv_radius       = 60.0/grid_res; // in grid squares
   conv_thresh       = >=5.0;
   vld_thresh        = 0.5;
   area_thresh       = >=0.0;
   inten_perc_value  = 100;
   inten_perc_thresh = >=0.0;
   merge_thresh      = >=1.25;
   merge_flag        = THRESH;
};
obs = fcst;
```

The **fcst_field** and **obs_field** variables specify the model variable and observation variable, respectively, and correspond 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 "**ANNN**", "**ZNNN**", "**PNNN**", "**PNNN-NNN**", "**LNNN**", or "**RNNN**" must follow each GRIB code. These indicate an accumulation interval, a single vertical level, a single pressure level, a range of pressure levels, a generic level, and a specific GRIB record number, respectively. "**NNN**" indicates the accumulation or level value. The value listed above indicates that accumulated precipitation with a 3-hourly accumulation interval should be verified.

The **raw_thresh** variable is used to threshold the raw forecast and / or observation 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.

The **conv_radius** variable defines the radius of the circular convolution applied to smooth the raw fields. The radii are specified in terms of grid units. The default convolution radii are defined in terms of the previously defined **grid_res** variable.

The **vld_thresh** 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 **vld_thresh** is set to 0.5.

The **conv_thresh** variable specifies 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.

The **area_thresh** variable specifies 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 that comprise it. A user may, for example, want to only consider objects that 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.

The **inten_perc_value** and **inten_perc_thresh** 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 that meet some maximum intensity criteria. By default, the intensity percentile threshold applied is greater than or equal to zero.

The **merge_thresh** 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 for both the forecast and observation fields. These thresholds should be chosen to define larger objects that fully contain the originally defined objects.

For example, for objects defined as **ge5.0**, a merge threshold of **ge2.5** will define larger objects that fully contain the original objects. Any two original objects contained within the same larger object will be merged. By default, the merge thresholds are set to be greater than or equal to 1.25.

The **merge_flag** variable controls what type of merging techniques will be applied to the objects defined in each field.

- **NONE** indicates that no merging should be applied.
- **THRESH** indicates that the double thresholding merging technique should be applied.
- **ENGINE** indicates that objects in each field should be merged by comparing the objects to themselves using a fuzzy engine approach.
- **BOTH** indicates that both techniques should be used.

By default, the double thresholding merging technique is applied.

```
mask_missing_flag = 0;
```

The **mask_missing_flag** variable specifies how missing data in the raw model and observation fields will be treated.

- **NONE** indicates no additional processing is to be done.
- **FCST** indicates missing data in the observation field should be used to mask the forecast field.
- **OBS** indicates missing data in the forecast field should be used to mask the observation field.
- **BOTH** indicates masking should be performed in both directions (i.e., mask the forecast field with the observation field and vice-versa).

Prior to defining objects, it is recommended that the raw fields be made to look similar to each other by assigning a value of 3 to this parameter. However, by default no masking is performed.

```
mask = {
   grid      = "";
   grid_flag = NONE; // Apply to NONE, FCST, OBS, or BOTH
   poly      = "";
   poly_flag = NONE; // Apply to NONE, FCST, OBS, or BOTH
};
```

The **grid** variable specifies a pre-defined NCEP grid with which to mask the raw forecast and observation fields. The predefined grids are specified as "**GNNN**" 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.

The **grid_flag** variable specifies how the **grid** should be applied.

- **NONE** indicates that the masking grid should not be applied.
- **FCST** indicates that the masking grid should be applied to the forecast field.
- **OBS** indicates that the masking grid should be applied to the observation field.
- **BOTH** indicates that the masking grid should be applied to both fields.

By default, the masking grid is not applied.

Similar to the **grid** variable above, the **poly** variable is the name of a file that defines a verification masking region. These masking regions may be specified in two ways: as a lat/lon polygon or using a gridded data file such as the NetCDF output of the Gen-Poly-Mask tool.

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.

Alternatively, any gridded data file that MET can read may be used to define a verification masking region. Users must specify a description of the field to be used from the input file and, optionally, may specify a threshold to be applied to that field. Any grid point where the resulting field is 0, the mask is turned off. Any grid point where it is non-zero, the mask is turned on.

Similar to the **grid_flag** variable above, the **poly_flag** variable specifies how the masking polygon should be applied.

- **NONE** indicates the masking polygon should be applied to neither field.
- **FCST** indicates the masking polygon should be applied to the forecast field.
- **OBS** indicates the masking polygon should be applied to the observation field.
- **BOTH** indicates that the masking polygon should be applied to both fields.

By default, the masking polygon is not applied.

---

match_flag = MERGE_BOTH;

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.

- **NONE** indicates that no matching should be performed between the fields at all.

- **MERGE_BOTH** indicates that additional merging is allowed in both fields.
- **MERGE_FCST** indicates that additional merging is allowed only in the forecast field.
- **NO_MERGE** indicates that no additional merging is allowed in either field, meaning that each object will match at most one object in the other field.

By default, additional merging is allowed in both fields.

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

Computing the attributes for all possible pairs of objects can take some time depending on the numbers of objects. The **max_centroid_dist** 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.

```
weight = {
   centroid_dist    = 2.0;
   boundary_dist    = 4.0;
   convex_hull_dist = 0.0;
   angle_diff       = 1.0;
   area_ratio       = 1.0;
   int_area_ratio   = 2.0;
   complexity_ratio = 0.0;
   inten_perc_ratio = 0.0;
   inten_perc_value = 50;
}
```

The **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.

The **inten_perc_value** variable corresponds to the **inten_perc_ratio**. The **inten_perc_value** 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.

```
interest_function = {

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

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

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

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

   corner   = 0.8;
   ratio_if = (
      (    0.0, 0.0 )
      ( corner, 1.0 )
      (    1.0, 1.0 )
   );

   area_ratio = ratio_if;

   int_area_ratio = (
      ( 0.00, 0.00 )
      ( 0.10, 0.50 )
      ( 0.25, 1.00 )
      ( 1.00, 1.00 )
   );

   complexity_ratio = ratio_if;

   inten_perc_ratio = ratio_if;
}
```

The 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.

---

total_interest_thresh = 0.7;

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

---

print_interest_thresh = 0.0;

The **print_interest_thresh** variable determines which pairs of object attributes will be written to the output object attribute ASCII file. The user may choose to set the **print_interest_thresh** to the same value as the **total_interest_thresh**, meaning that only object pairs that actually match are written to the output file. 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_plot = {
   color_table      =
"MET_BASE/data/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
   colorbar_spacing = 1;
};
```

```
obs_raw_plot = {
   color_table      =
"MET_BASE/data/colortables/met_default.ctable";
   plot_min         = 0.0;
   plot_max         = 0.0;
   colorbar_spacing = 1;
};

object_plot = {
   color_table      =
"MET_BASE/data/colortables/mode_obj.ctable";
};
```

The **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** group and **obs_raw** group, while the objects identified are plotted using the values in the **object_plot** group. 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.

The **plot_min** and **plot_max** 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.

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

```
ps_plot_flag    = TRUE;
nc_pairs_flag   = TRUE;
ct_stats_flag   = TRUE;
```

These flags can be set to TRUE or FALSE to produce additional output, in the form of postscript plots (**ps_plot_flag**), NetCDF paris data(**nc_pairs_flag)**, or contingency table counts and statistics (**ct_stats_flag**).

```
output_prefix = "";
```

This option specifies a string to be used in the output file name. It can be useful for keeping results for different models or variables from overwriting each other.

```
version = "V4.0";
```

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`_PREFIX_FCST_VAR_LVL_vs_ OBS_VAR_LVL_HHMMSS`L`_YYYYMMDD_HHMMSS`V`_ HHMMSS`A`_`cts.txt` where **PREFIX** indicates the user-defined output prefix, **FCST_VAR_LVL** is the forecast variable and vertical level being used, **OBS_VAR_LVL** is the observation variable and vertical level being used, **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	VERSION	Version number (set to 3.0)
2	MODEL	User provided text string designating model name
3	FCST_LEAD	Forecast lead time in HHMMSS format
4	FCST_VALID	Forecast valid start time in YYYYMMDDHH format
5	FCST_ACCUM	Forecast accumulation time in HHMMSS format
6	OBS_LEAD	Observation lead time in HHMMSS format; when field2 is actually an observation, this should be "000000"
7	OBS_VALID	Observation valid start time in YYYYMMDDHH format
8	OBS_ACCUM	Observation accumulation time in HHMMSS format
9	FCST_RAD	Forecast convolution radius in grid squares
10	FCST_THR	Forecast convolution threshold
11	OBS_RAD	Observation convolution radius in grid squares
12	OBS_THR	Observation convolution threshold
13	FCST_VAR	Forecast variable
14	FCST_LEV	Forecast vertical level

MODE ASCII CONTINGENCY TABLE OUTPUT FORMAT		
Column Number	**MODE CTS Column Name**	**Description**
15	OBS_VAR	Observation variable
16	OBS_LEV	Observation vertical level
17 FIELD		Field type for this line: • RAW for the raw input fields • FILTER for the raw fields after applying the raw thresholds • OBJECT for the resolved object fields
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 cluster objects, and pairs of objects. Each line in this file contains the same number of columns, though those columns not applicable to a given line contain fill data. The first row of every MODE object attribute file is a header containing the column names. The number of lines in this file depends on the number of objects defined. This file contains lines of 6 types that are indicated by the contents of the "**object_id**" column. The "**object_id**" can take the following 6 forms: **FNNN**, **oNNN**, **FNNN_oNNN**, **CFNNN**, **coNNN**, **CFNNN_coNNN**. In each case, **NNN** is a three-digit number indicating the object index. While all lines have the first 18 header columns in common, these 6 forms for "**object_id**" can be divided into two types – one for single objects and one for pairs of objects. The single object lines (**FNNN**, **oNNN**, **CFNNN**, and **coNNN**) contain valid data in columns 19-39 and fill data in columns 40-51. The object pair lines (**FNNN_oNNN** and **CFNNN_coNNN**) contain valid data in columns 40-51 and fill data in columns 19-39. These object identifiers are described in Table 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	1-18, 19-39	Attributes for simple forecast, observation objects
FNNN_ONNN	1-18, 40-51	Attributes for pairs of simple forecast and observation objects
CFNNN, CONNN	1-18, 19-39	Attributes for merged cluster objects in forecast, observation fields
CFNNN_CONNN	1-18, 40-51	Attributes for pairs of forecast, observation cluster objects

A note on terminology: a cluster (referred to as "composite" in earlier versions) object need not necessarily consist of more than one simple object. A cluster object is by definition any set of one or more objects in one field which match a set of one or more objects in the other field. When a single simple forecast object matches a single simple observation object, they are each considered to be cluster objects as well.

The contents of the columns in this ASCII file are summarized in Table 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	VERSION	Version number (set to 3.0)
2	MODEL	User provided text string designating model name
3	FCST_LEAD	Forecast lead time in HHMMSS format
4	FCST_VALID	Forecast valid start time in YYYYMMDDHH format
5	FCST_ACCUM	Forecast accumulation time in HHMMSS format
6	OBS_LEAD	Observation lead time in HHMMSS format; when field2 is actually an observation, this should be "000000"
7	OBS_VALID	Observation valid start time in YYYYMMDDHH format
8	OBS_ACCUM	Observation accumulation time in HHMMSS format
9	FCST_RAD	Forecast convolution radius in grid squares
10	FCST_THR	Forecast convolution threshold
11	OBS_RAD	Observation convolution radius in grid squares
12	OBS_THR	Observation convolution threshold
13	FCST_VAR	Forecast variable

MODE ASCII OBJECT ATTRIBUTE OUTPUT FORMAT		
Column Number	**MODE Column Name**	**Description**
14	FCST_LEV	Forecast vertical level
15	OBS_VAR	Observation variable
16	OBS_LEV	Observation vertical level
17	OBJECT_ID	Object numbered from 1 to the number of objects in each field
18	OBJECT_CAT	Object category indicating to which cluster object it belongs
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	AREA_THRESH	Area of the object containing data values in the filtered field that meet the object definition threshold criteria (in grid squares)
29	CURVATURE	Radius of curvature of the object defined in terms of third order moments (in grid units)
30-31	CURVATURE_X, _Y	Center of curvature (in grid coordinates)
32	COMPLEXITY	Ratio of the area of an object to the area of its convex hull (unitless)
33-37	INTENSITY_10, _25, _50, _75, _90	10^{th}, 25^{th}, 50^{th}, 75^{th}, and 90^{th} percentiles of intensity of the filtered field within the object (various units)
38	INTENSITY_NN	The percentile of intensity chosen for use in the percentile intensity ratio (column 50; variable units)
39	INTENSITY_SUM	Sum of the intensities of the filtered field within the object (variable units)
40	CENTROID_DIST	Distance between two objects centroids (in grid units)
41	BOUNDARY_DIST	Minimum distance between the boundaries of two objects (in grid units)
42	CONVEX_HULL_DIST	Minimum distance between the convex hulls of two objects (in grid units)
43	ANGLE_DIFF	Difference between the axis angles of two objects (in degrees)
44	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)

MODE ASCII OBJECT ATTRIBUTE OUTPUT FORMAT		
Column Number	**MODE Column Name**	**Description**
45	INTERSECTION_AREA	Intersection area of two objects (in grid squares)
46	UNION_AREA	Union area of two objects (in grid squares)
47	SYMMETRIC_DIFF	Symmetric difference of two objects (in grid squares)
48	INTERSECTION_OVER_AREA	Ratio of intersection area to union area (unitless)
49	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)
50	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)
51	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 cluster forecast objects, and indices for the matched cluster observation objects. The NetCDF file also contains lat/lon and x/y data for the vertices of the polygons for the boundaries of the simple forecast and observation objects. The generation of this file can be disabled using the **`–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)
fcst_obj	Number of simple forecast objects
fcst_bdy	Number of points used to define the boundaries of all of the simple forecast objects
obs_obj	Number of simple observation objects
obj_bdy	Number of points used to define the boundaries of all of the simple observation objects

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

`mode` *NetCDF OUTPUT FILE VARIABLES*		
NetCDF Variable	**Dimension**	**Description**
fcst_obj_id	lat, lon	Simple forecast object id number for each grid point
fcst_comp_id	lat, lon	Cluster 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	Cluster observation object id number for each grid point
n_fcst_obj	-	Number of simple forecast objects
n_obs_obj	-	Number of simple observation objects
fcst_obj_bdy_start	fcst_obj	Index into the forecast boundary lat/lon and x/y arrays for this object
fcst_obj_bdy_npts	fcst_obj	Number of boundary points in the lat/lon and x/y arrays for this object
fcst_bdy_lat	fcst_bdy	Latitude value for the forecast boundary points
fcst_bdy_lon	fcst_bdy	Longitude value for the forecast boundary points
fcst_bdy_x	fcst_bdy	Grid-x value for the forecast boundary points
fcst_bdy_y	fcst_bdy	Grid-y value for the forecast boundary points
obs_obj_bdy_start	obs_obj	Index into the observation boundary lat/lon and x/y arrays for this object
obs_obj_bdy_npts	obs_obj	Number of boundary points in the lat/lon and x/y arrays for this object
obs_bdy_lat	obs_bdy	Latitude value for the observation boundary points
obs_bdy_lon	obs_bdy	Longitude value for the observation boundary points
obs_bdy_x	obs_bdy	Grid-x value for the observation boundary points
obs_bdy_y	obs_bdy	Grid-y value for the observation boundary points

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 5 summary pages at a minimum, but the number of pages will depend on the merging options chosen. Additional pages will be created if merging is performed using the double thresholding or fuzzy engine merging techniques for the forecast and/or observation fields. Examples of the PostScript plots can be obtained by running the example cases provided with the MET tarball.

The first page of PostScript output contains a great deal of summary information. Six tiles of images provide thumbnail images of the raw fields, matched/merged object fields, and object index fields for the forecast and observation grids. In the matched/merged object fields, matching colors of objects across fields indicate that the corresponding objects match, while within a single field, black outlines indicate merging. Note that objects that are colored royal blue are unmatched. Along the bottom of the

page, the criteria used for object definition and matching/merging are listed. Along the right side of the page, total interest values for pairs of simple objects are listed in sorted order. The numbers in this list correspond to the object indices shown in the object index plots.

The second and third pages of the PostScript output file display enlargements of the forecast and observation raw and object fields, respectively. The fourth page displays the forecast object with the outlines of the observation objects overlaid, and vice versa. The fifth page contains summary information about the pairs of matched cluster objects.

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

Chapter 7 – The Wavelet-Stat Tool

7.1 Introduction

The Wavelet-Stat tool decomposes two-dimensional forecasts and observations according to intensity and scale. This chapter provides a description of the MET Wavelet-Stat Tool, which enables users to apply the Intensity-Scale verification technique described by Casati et al. (2004).

The Intensity-Scale technique is one of the recently developed verification approaches that focus on verification of forecasts defined over spatial domains. Spatial verification approaches, as opposed to point-by-point verification approaches, aim to account for the presence of features and for the coherent spatial structure characterizing meteorological fields. Since these approaches account for the intrinsic spatial correlation existing between nearby grid-points, they do not suffer from point-by-point comparison related verification issues, such as double penalties. Spatial verification approaches aim to account for the observation and forecast time-space uncertainties, and aim to provide feedback on the forecast error in physical terms.

The Intensity-Scale verification technique, as most of the spatial verification approaches, compares a forecast field to an observation field. To apply the Intensity-Scale verification approach, observations need to be defined over the same spatial domain of the forecast to be verified.

Within the spatial verification approaches, the Intensity-Scale technique belongs to the scale-decomposition (or scale-separation) verification approaches. The scale-decomposition approaches enable users to perform the verification on different spatial scales. Weather phenomena on different scales (e.g. frontal systems versus convective showers) are often driven by different physical processes. Verification on different spatial scales can therefore provide deeper insights into model performance at simulating these different processes.

The spatial scale components are obtained usually by applying a single band spatial filter to the forecast and observation fields (e.g. Fourier, Wavelets). The scale-decomposition approaches measure error, bias and skill of the forecast on each different scale component. The scale-decomposition approaches therefore provide feedback on the scale dependency of the error and skill, on the no-skill to skill transition scale, and on the capability of the forecast of reproducing the observed scale structure.

The Intensity-Scale technique evaluates the forecast skill as a function of the intensity values and of the spatial scale of the error. The scale components are obtained by applying a two dimensional Haar wavelet filter. Note that wavelets, because of their locality, are suitable for representing discontinuous fields characterized by few sparse non-zero features, such as precipitation. Moreover, the technique is based on a categorical approach, which is a robust and resistant approach, suitable for non-normally distributed variables, such as precipitation. The intensity-scale technique was

specifically designed to cope with the difficult characteristics of precipitation fields, and for the verification of spatial precipitation forecasts. However, the intensity-scale technique can also be applied to verify other variables, such as cloud fraction.

7.2 Scientific and statistical aspects

7.2.1 The method

Casati et al (2004) applied the Intensity-Scale verification to preprocessed and re-calibrated (unbiased) data. The preprocessing was aimed to mainly normalize the data, and defined categorical thresholds so that each categorical bin had a similar sample size. The recalibration was performed to eliminate the forecast bias. Preprocessing and recalibration are not strictly necessary for the application of the Intensity-Scale technique. The MET Intensity-Scale Tool does not perform either, and applies the Intensity-Scale approach to biased forecasts, for categorical thresholds defined by the user.

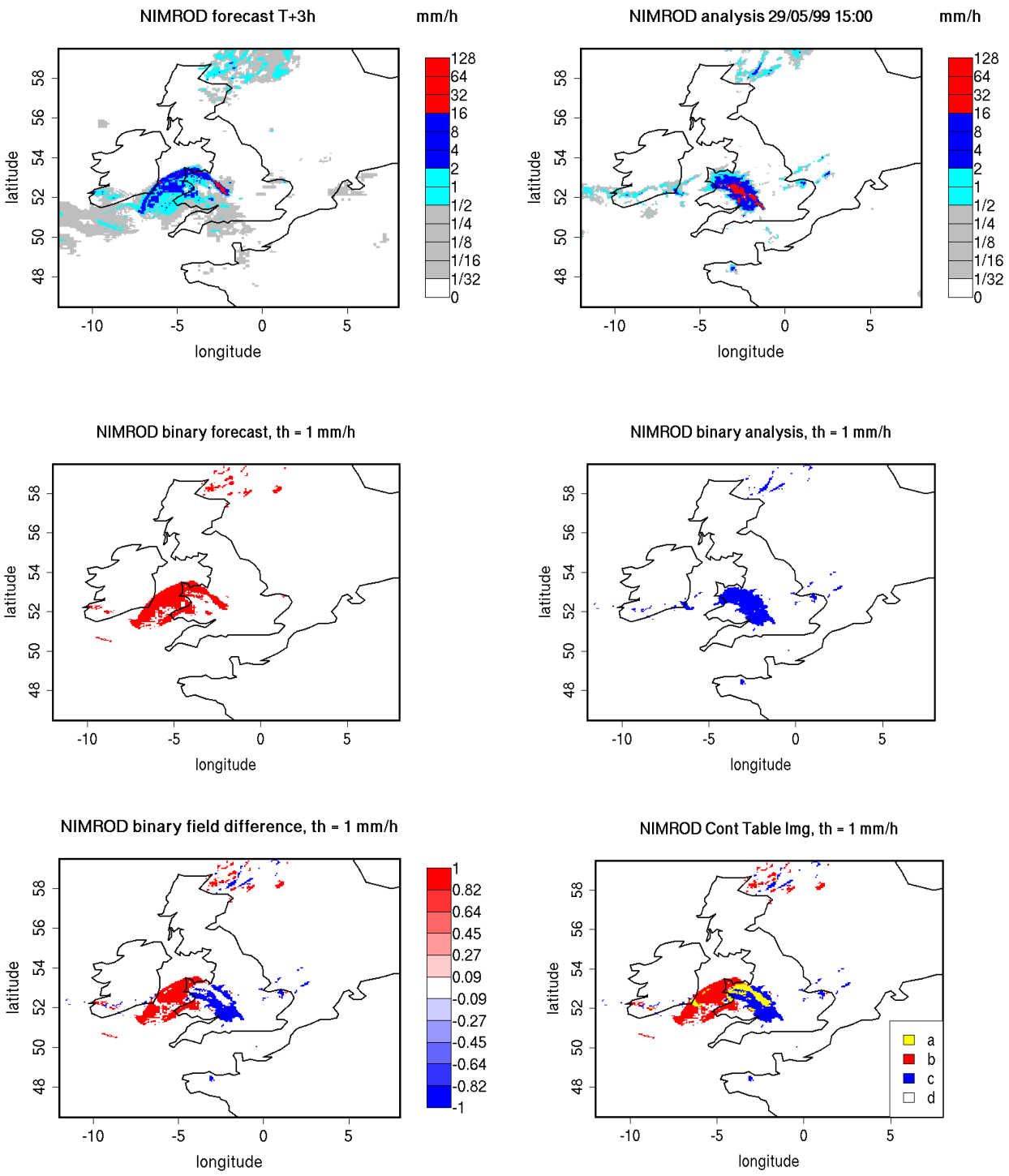


Figure 7.1: *NIMROD 3h lead-time forecast and corresponding verifying analysis field (precipitation rate in mm/h, valid the 05/29/99 at 15:00 UTC); forecast and analysis binary fields obtained for a threshold of 1mm/h, the binary field difference has their corresponding Contingency Table Image (see Table 7.1). The forecast shows a storm of 160 km displaced almost its entire length.*

The Intensity Scale approach can be summarized in the following 5 steps:

1. For each threshold, the forecast and observation fields are transformed into binary fields: where the grid-point precipitation value meets the threshold criteria it is assigned 1, where the threshold criteria are not met it is assigned 0. Figure 7.1 illustrates an example of a forecast and observation fields, and their corresponding binary fields for a threshold of 1mm/h. This case shows an intense storm of the scale of 160 km displaced almost its entire length. The displacement error is clearly visible from the binary field difference and the contingency table image obtained for the same threshold (Table 7.1).

2. The binary forecast and observation fields obtained from the thresholding are then decomposed into the sum of components on different scales, by using a 2D Haar wavelet filter (Fig 7.2). Note that the scale components are fields, and their sum adds up to the original binary field. For a forecast defined over square domain of 2^n x 2^n grid-points, the scale components are n+1: n mother wavelet components + the largest father wavelet (or scale-function) component. The n mother wavelet components have resolution equal to 1, 2, 4, ... 2^{n-1} grid-points. The largest father wavelet component is a constant field over the 2^n x 2^n grid-point domain with value equal to the field mean.
   Note that the wavelet transform is a linear operator: this implies that the difference of the spatial scale components of the binary forecast and observation fields (Fig 7.2) are equal to the spatial scale components of the difference of the binary forecast and observation fields (Fig. 7.3), and these scale components also add up to the original binary field difference (Fig. 7.1). The intensity-scale technique considers thus the spatial scale of the error. For the case illustrated (Fig 7.1 and Fig 7.3) note the large error associated at the scale of 160 km, due the storm, 160km displaced almost its entire length.
   Note also that the means of the binary forecast and observation fields (i.e. their largest father wavelet components) are equal to the proportion of forecast and observed events above the threshold, (a+b)/n and (a+c)/n, evaluated from the contingency table counts (Table 7.1) obtained from the original forecast and observation fields by thresholding with the same threshold used to obtained the binary forecast and observation fields. This relation is intuitive when observing forecast and observation binary fields and their corresponding contingency table image (Fig 7.1). The comparison of the largest father wavelet component of binary forecast and observation fields therefore provides feedback on the whole field bias.

3. For each threshold (t) and for each scale component (j) of the binary forecast and observation, the Mean Squared Error (MSE) is then evaluated (Figure 7.4). The error is usually large for small thresholds, and decreases as the threshold increases. This behavior is partially artificial, and occurs because the smaller the threshold the more events will exceed it, and therefore the larger would be the error, since the error tends to be proportional to the amount of events in the

binary fields. The artificial effect can be diminished by normalization: because of the wavelet orthogonal properties, the sum of the MSE of the scale components is equal to the MSE of the original binary fields: $MSE(t) = \Sigma_j MSE(t,j)$. Therefore, the percentage that the MSE for each scale contributes to the total MSE may be computed: for a given threshold, t, $MSE\%(t,j) = MSE(t,j)/MSE(t)$. The MSE% does not exhibit the threshold dependency, and usually shows small errors on large scales and large errors on small scales, with the largest error associated to the smallest scale and highest threshold. For the NIMROD case illustrated note the large error at 160 km and between the thresholds of ½ and 4 mm/h, due to the storm, 160km displaced almost its entire length.

Note that the MSE of the original binary fields is equal to the proportion of the counts of misses (c/n) and false alarms (b/n) for the contingency table (Table 71) obtained from the original forecast and observation fields by thresholding with the same threshold used to obtain the binary forecast and observation fields: $MSE(t) = (b+c)/n$. This relation is intuitive when comparing the forecast and observation binary field difference and their corresponding contingency table image (Fig 71).

4. The MSE for the random binary forecast and observation fields is estimated by $MSE(t)_{random}=FBI*Br*(1-Br) + Br*(1-FBI*Br)$, where $FBI=(a+b)/(a+c)$ is the frequency bias index and $Br=(a+c)/n$ is the sample climatology from the contingency table (Table 7.1) obtained from the original forecast and observation fields by thresholding with the same threshold used to obtain the binary forecast and observation fields. This formula follows by considering the Murphy and Winkler (1987) framework, appling the Bayes' theorem to express the joint probabilities b/n and c/n as product of the marginal and conditional probability (e.g. Jolliffe and Stephenson, 2003; Wilks, 2006), and then noticing that for a random forecast the conditional probability is equal to the unconditional one, so that b/n and c/n are equal to the product of the corresponding marginal probabilities solely.

5. For each threshold (t) and scale component (j), the skill score based on the MSE of binary forecast and observation scale components is evaluated (Figure 7.5). The standard skill score definition as in Jolliffe and Stephenson (2003) or Wilks (2006) is used, and random chance is used as reference forecast. The MSE for the random binary forecast is equipartitioned on the n+1 scales to evaluate the skill score: $SS(t,j)=1-MSE(t,j)*(n+1)/MSE(t)_{random}$

   The Intensity-Scale (IS) skill score evaluates the forecast skill as a function of the precipitation intensity and of the spatial scale of the error. Positive values of the IS skill score are associated to a skillful forecast, whereas negative values are associated to no skill. Usually large scales exhibit positive skill (large scale events, such as fronts, are well predicted), whereas small scales exhibit negative skill (small scale events, such as convective showers, are less predictable), and the smallest scale and highest thresholds exhibit the worst skill. For the NIMROD case illustrated note the negative skill associated to the 160 km scale, for the thresholds ½ to 4 mm/h, due to the 160 km storm displaced almost its entire length.

***Table 7-1**: 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")**	*Hits = a*	*False Alarms = b*	*a + b*
***f* = 0 (e.g., "No")**	*Misses = c*	*Correct rejections = d*	*c + d*
Total	*a + c*	*b + d*	*a + b + c + d*

In addition to the MSE and the SS, the energy squared is also evaluated, for each threshold and scale (Fig 7.6). The energy squared of a field X is the average of the squared values: $En2(X)= \Sigma_i x_i^2$. The energy squared provides feedback on the amount of events present in the forecast and observation fields for each scale, for a given threshold. Usually, small thresholds are associated to a large energy, since many events exceed the threshold. Large thresholds are associated to a small energy, since few events exceed the threshold. Comparison of the forecast and observed squared energy provide feedback on the bias on different scales, for each threshold.

The En2 bias for each threshold and scale is assessed by the En2 relative difference, equal to the difference between forecast and observed squared energies normalized by their sum: [En2(F)-En2(O)]/[En2(F)+End2(O)]. Since defined in such a fashion, the En2 relative difference accounts for the difference between forecast and observation squared energies relative to their magnitude, and it is sensitive therefore to the ratio of the forecast and observed squared energies. The En2 relative difference ranges between -1 and 1, positive values indicate over-forecast and negative values indicate under-forecast. For the NIMROD case illustrated the forecast exhibits over-forecast for small thresholds, quite pronounced on the large scales, and under-forecast for high thresholds.

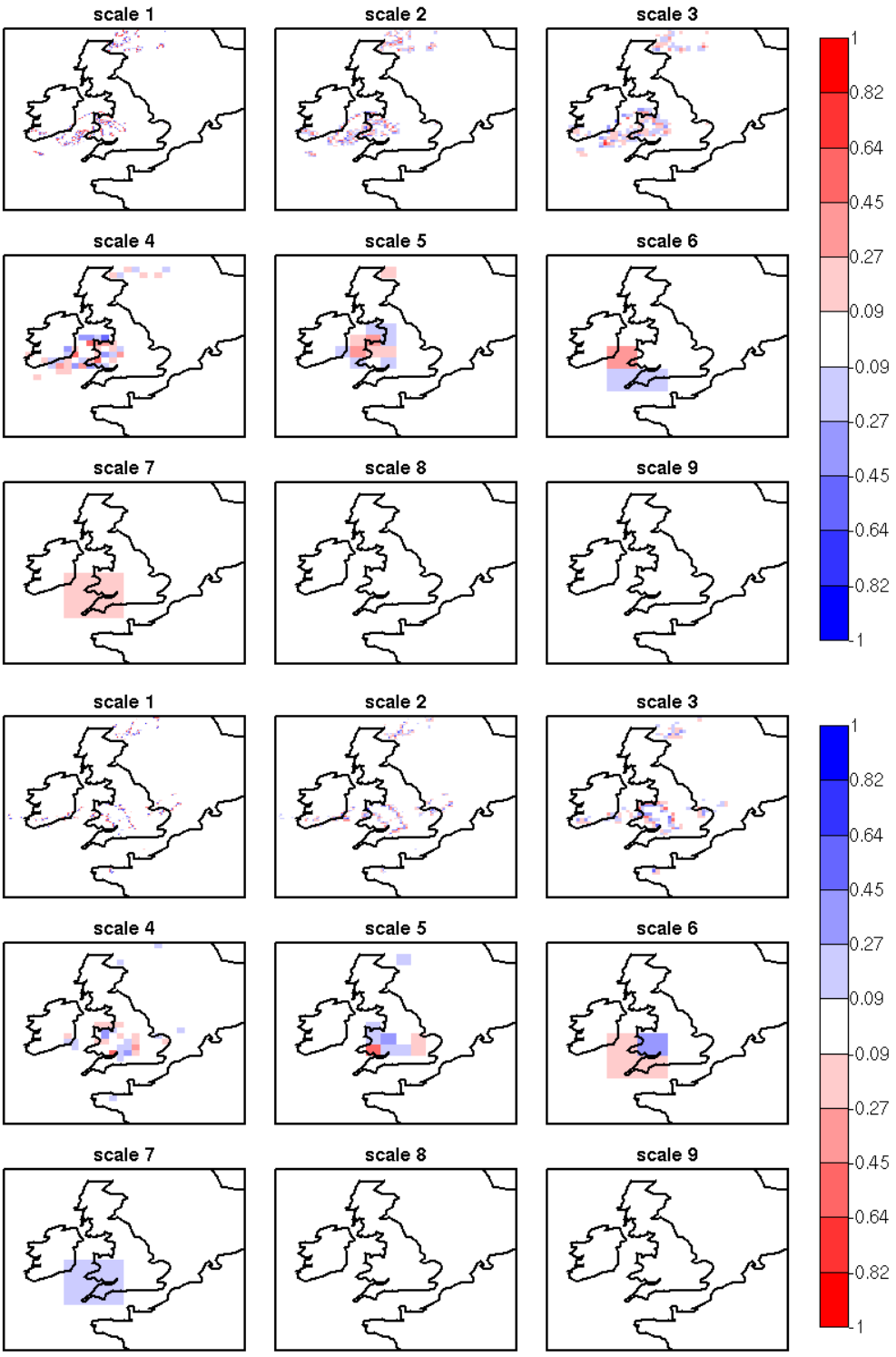


Figure 7.2: *NIMROD binary forecast (top) and binary analysis (bottom) spatial scale components obtained by a 2D Haar wavelet transform (th=1 mm/h). Scale 1 to 8 refer to mother wavelet components (5, 10, 20, 40, 80, 160, 320, 640 km resolution); scale 9 refer to the largest father wavelet component (1280 km resolution)*

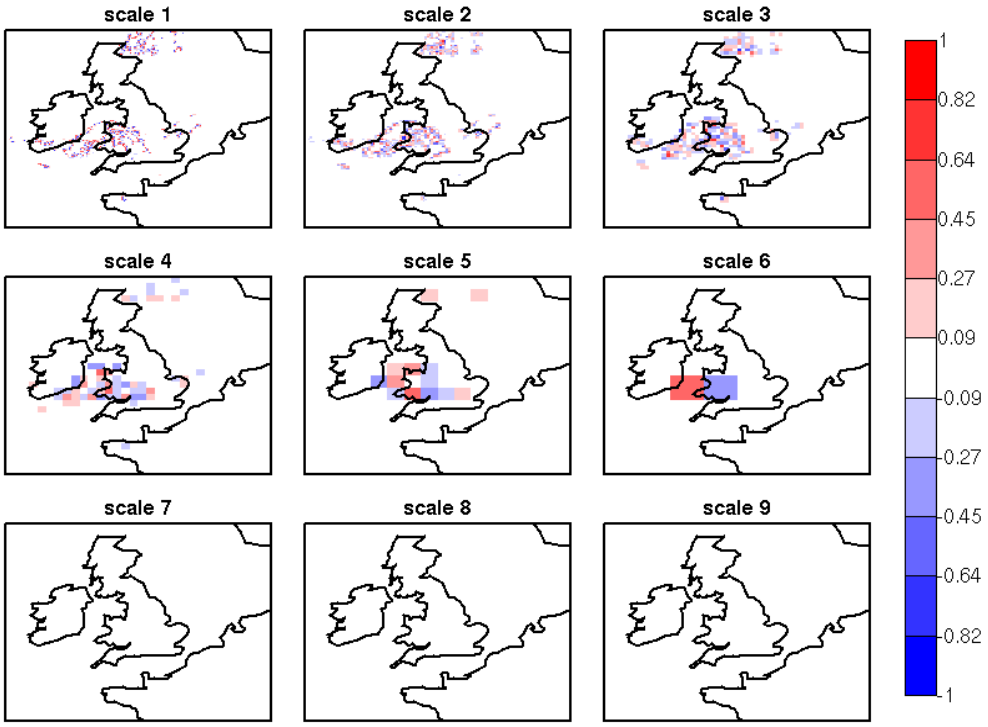


Figure 7.3: *NIMROD binary field difference spatial scale components obtained by a 2D Haar wavelet transform (th=1 mm/h). Scales 1 to 8 refer to mother wavelet components (5, 10, 20, 40, 80, 160, 320, 640 km resolution); scale 9 refers to the largest father wavelet component (1280 km resolution). Note the large error at the scale 6 = 160 km, due to the storm, 160 km displaced almost of its entire length.*

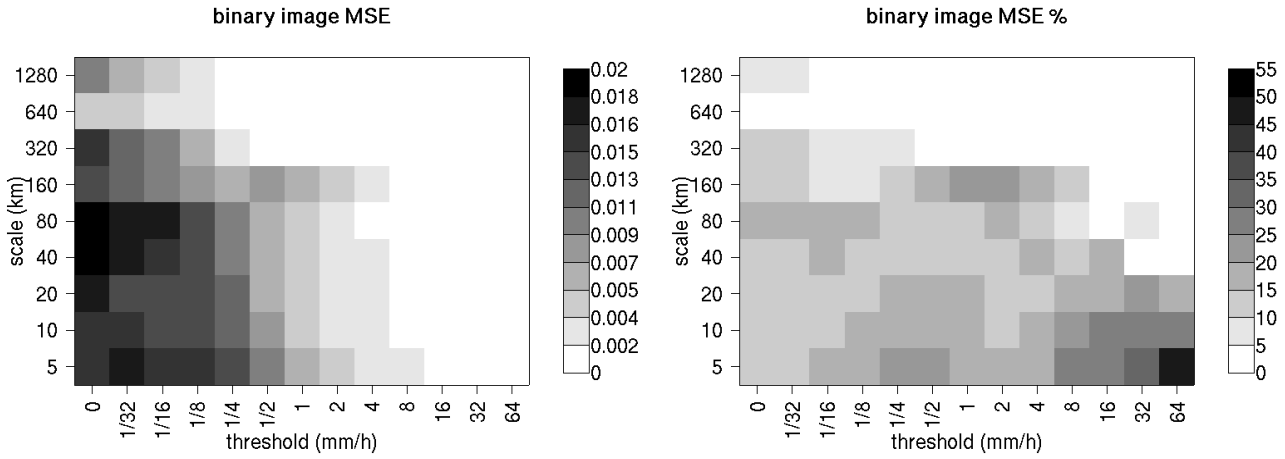


Figure 7.4: *MSE and MSE % for the NIMROD binary forecast and analysis spatial scale components. In the MSE%, note the large error associated to the scale 6 = 160 km, for the thresholds ½ to 4 mm/h, associated to the displaced storm.*

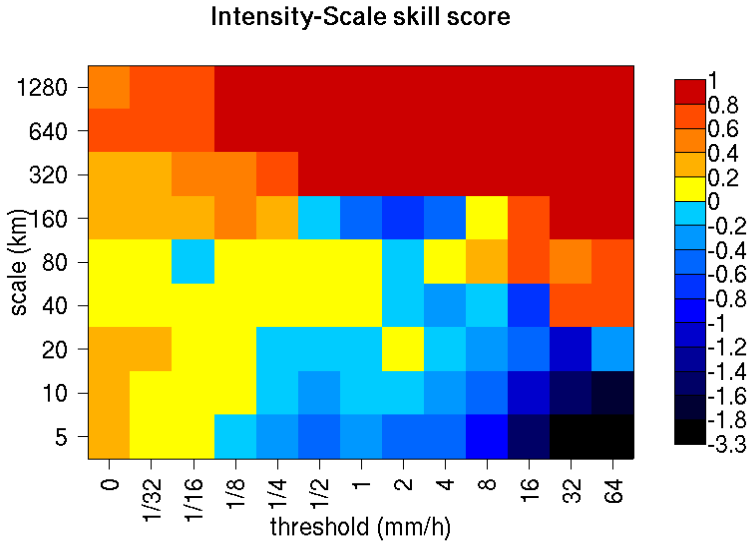


Figure 7.5: *Intensity-Scale skill score for the NIMROD forecast and analysis shown in Fig 7.1. The skill score is a function of the intensity of the precipitation rate and spatial scale of the error. Note the negative skill associated to the scale 6 = 160 km, for the thresholds ½ to 4 mm/h, associated to the displaced storm.*

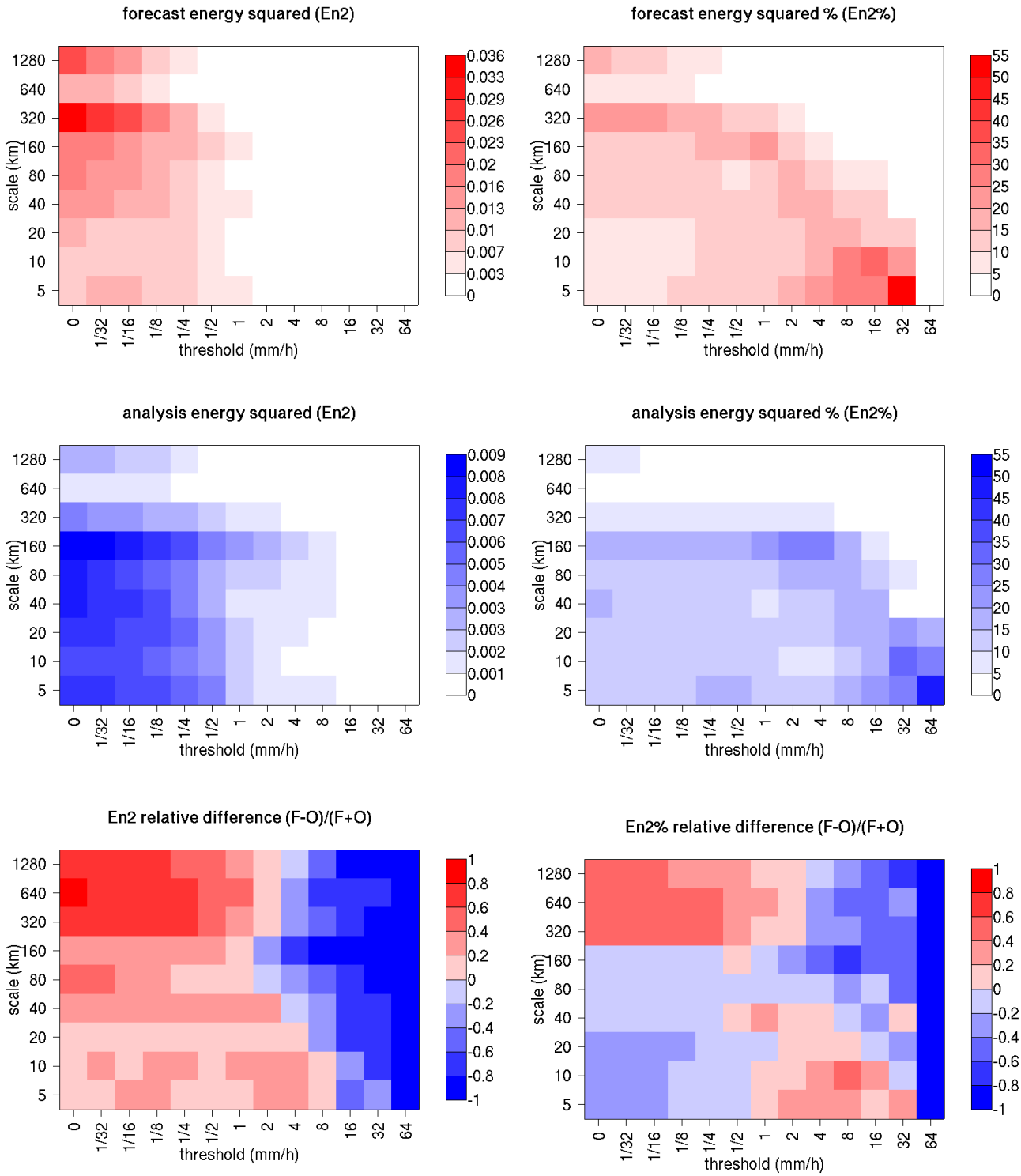


Figure 7.6: *Energy squared and energy squared percentages, for each threshold and scale, for the NIMROD forecast and analysis, and forecast and analysis En2 and En2% relative differences.*

As for the MSE, the sum of the energy of the scale components is equal to the energy of the original binary field: $En2(t) = \Sigma_j En2(t,j)$. Therefore, the percentage that the En2 for each scale contributes the total En2 may be computed: for a given threshold, t, $En2\%(t,j) = En2(t,j)/En2(t)$. Usually, for precipitation fields, low thresholds exhibit most of the energy percentage on large scales (and less percentage on the small scales), since low thresholds are associated to large scale features, such as fronts. On the other hand, for higher thresholds the energy percentage is usually larger on small scales, since intense events are associated to small scales features, such as convective cells or showers. The comparison of the forecast and observation squared energy percentages provides feedback on how the events are distributed across the scales, and enable the comparison of forecast and observation scale structure.

For the NIMROD case illustrated, the scale structure is assessed again by the relative difference, but calculated of the squared energy percentages. For small thresholds the forecast over-estimates the number of large scale events and under-estimates the number of small scale events, in proportion to the total number of events. On the other hand, for larger thresholds the forecast under-estimates the number of large scale events and over-estimates the number of small scale events, again in proportion to the total number of events. Overall it appears that the forecast over-estimates the percentage of events associated to high occurrence, and under-estimate the percentage of events associated to low occurrence. The En2% for the 64 mm/h thresholds is homogeneously under-estimated for all the scales, since the forecast does not have any event exceeding this threshold.

Note that the energy squared of the observation binary field is identical to the sample climatology Br=(a+c)/n. Similarly, the energy squared of the forecast binary field is equal to (a+b)/n. The ratio of the squared energies of the forecast and observation binary fields is equal to the FBI=(a+b)/(a+c), for the contingency table (Table 7.1) obtained from the original forecast and observation fields by thresholding with the same threshold used to obtained the binary forecast and observation fields.

7.2.2 The spatial domain constraints

The Intensity-Scale technique is constrained by the fact that orthogonal wavelets (discrete wavelet transforms) are usually performed on square domains of 2^n x 2^n grid-points. The Wavelet-Stat tool handles this issue based on settings in the configuration file. The Wavelet-Stat tool can define tiles of dimensions 2^n x 2^n over the input domain in the following ways:

1. User-Defined Tiling: The user may define one or more tiles of size 2^n x 2^n over

their domain to be applied. This is done by selecting the grid coordinates for the lower-left corner of the tile(s) and the tile dimension to be used. If the user specifies more than one tile, the Intensity-Scale method will be applied to each tile separately. At the end, the results will automatically be aggregated across all the tiles and written out with the results for each of the individual tiles. Users are encouraged to select tiles which consist entirely of valid data.

2. Automated Tiling: This tiling method is essentially the same as the user-defined tiling method listed above except that the tool automatically selects the location and size of the tile(s) to be applied. It figures out the maximum tile of dimension 2^n x 2^n that fits within the domain and places the tile at the center of the domain. For domains that are very elongated in one direction, it defines as many of these tiles as possible that fit within the domain.

3. Padding: If the domain size is only slightly smaller than 2^n x 2^n, for certain variables (e.g. precipitation), it is advisable to expand the domain out to 2^n x 2^n grid-points by adding extra rows and/or columns of fill data. For precipitation variables, a fill value of zero is used. For continuous variables, such as temperature, the fill value is defined as the mean of the valid data in the rest of the field. A drawback to the padding method is the introduction of artificial data into the original field. Padding should only be used when a very small number of rows and/or columns need to be added.

7.2.3 Aggregation of statistics on multiple cases

The Stat-Analysis tool aggregates the intensity scale technique results. Since the results are scale-dependent, it is sensible to aggregate results from multiple model runs (e.g. daily runs for a season) on the same spatial domain, so that the scale components for each singular case will be the same number, and the domain, if not a square domain of 2^n x 2^n grid-points, will be treated in the same fashion. Similarly, the intensity thresholds for each run should all be the same.

The MSE and forecast and observation squared energy for each scale and thresholds are aggregated simply with a weighted average, where weights are proportional to the number of grid-points used in each single run to evaluate the statistics. If the same domain is always used (and it should) the weights result all the same, and the weighted averaging is a simple mean. For each threshold, the aggregated Br is equal to the aggregated squared energy of the binary observation field, and the aggregated FBI is obtained as the ratio of the aggregated squared energies of the forecast and observation binary fields. From aggregated Br and FBI, the MSE_{random} for the aggregated runs can be evaluated using the same formula as for the single run. Finally, the Intensity-Scale Skill Score is evaluated by using the aggregated statistics within the same formula used for the single case.

7.3 Practical information

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

7.3.1 `Wavelet_stat` *usage*

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

```
Usage: wavelet_stat
     fcst_file
     obs_file
     config_file
     [-outdir path]
     [-log file]
     [-v level]
```

`Wavelet_stat` has three required arguments and accepts up to seven optional ones.

***Required arguments for* `wavelet_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* `wavelet_stat`**

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

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

3. The **`-v` level** option indicates the desired level of verbosity. The contents of "**level**" will override the default setting of 2. Setting the verbosity to 0 will make

the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

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

Example:

```
wavelet_stat    sample_fcst.grb
                sample_obs.grb
                WaveletStatConfig
```

In the example, the Wavelet-Stat tool will verify the model data in the **`sample_fcst.grb`** GRIB file using the observations in the **`sample_obs.grb`** GRIB file applying the configuration options specified in the **`WaveletStatConfig`** file.

7.3.2 `wavelet_stat` *configuration file*

The default configuration file for the Wavelet-Stat tool, **`WaveletStatConfig_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 **`WaveletStatConfig_default`** and the required variables for any `wavelet_stat` 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 tool.

```
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".

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

```
      }
   ];
};
obs = fcst;
```

The **fcst** and **obs** groups specify one or more data fields to be selected from the forecast and observation files to be compared. The fields should be specified in the same way that they're specified for the other MET tools, with name, level and categorical thresholds.

```
mask_missing_flag = NONE;
```

The **mask_missing_flag** variable specifies how missing data in the raw model and observation fields will be treated.

- **NONE** indicates no additional processing is to be done.
- **FCST** indicates missing data in the observation field should be used to mask the forecast field.
- **OBS** indicates missing data in the forecast field should be used to mask the observation field.
- **BOTH** indicates masking should be performed in both directions (i.e., mask the forecast field with the observation field and vice-versa).

```
grid_decomp_flag = AUTO;

tile = {
   width = 0;
   location = [
      {
         x_ll = 0;
         y_ll = 0;
      }
   ];
};
```

The **grid_decomp_flag** variable specifies how tiling should be performed:

- **AUTO** indicates that the automated-tiling should be done.
- **TILE** indicates that the user-defined tiles should be applied.
- **PAD** indicated that the data should be padded out to the nearest dimension of 2^n x 2^n

The **xll** and **yll** variables allow users to manually define the tiles of dimension they would like to apply. The **tile_xll** and **tile_yll** variables specify the location of one or more lower-left tile grid (x, y) points.

```
wavelet = {
  type   = HAAR;
  member = 2;
};
```

The **wavelet_flag** and **wavelet_k** variables specify the type and shape of the wavelet to be used for the scale decomposition. The Casati et al. (2004) method uses a Haar wavelet which is a good choice for discontinuous fields like precipitation. However, users may choose to apply any wavelet family/shape that is available in the GNU Scientific Library. Values for the **wavelet_flag** variable, and associated choices for k, are described below:

- **HAAR** for the Haar wavelet (member = 2).
- **HAAR_CNTR** for the Centered-Haar wavelet (member = 2).
- **DAUB** for the Daubechies wavelet (member = 4, 6, 8, 10, 12, 14, 16, 18, 20).
- **DAUB_CNTR** for the Centered-Daubechies wavelet (member = 4, 6, 8, 10, 12, 14, 16, 18, 20).
- **BSPLINE** for the Bspline wavelet (member = 103, 105, 202, 204, 206, 208, 301, 303, 305, 307, 309).
- **BSPLINE_CNTR** for the Centered-Bspline wavelet (member = 103, 105, 202, 204, 206, 208, 301, 303, 305, 307, 309).

```
output_flag = {
   isc = BOTH;
};
```

The **output_flag** array controls the type of output that the Wavelet-Stat tool generates. These flags are set similarly to the output flags of the other MET tools, with possible values of NONE, STAT, and BOTH. The isc line type for Intensity-Scale STAT lines.

```
nc_pairs_flag = TRUE;
ps_plot_flag  = TRUE;
```

These flags produce additional output from the Wavelet-Stat tool. The other two types of output from the wavelet tool are:

1. NetCDF output file containing the scale decompositions for each combination of tile/field/threshold.
2. PostScript output file summarizing the Intensity-Scale method.

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

The Wavelet-Stat 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_plot = {
   color_table = "MET_BASE/data/colortables/met_default.ctable";
   plot_min = 0.0;
   plot_max = 0.0;
};

obs_raw_plot = {
   color_table = "MET_BASE/data/colortables/met_default.ctable";
   plot_min = 0.0;
   plot_max = 0.0;
};

wvlt_plot = {
   color_table =
"MET_BASE/data/colortables/NCL_colortables/BlWhRe.ctable";
   plot_min = -1.0;
   plot_max =  1.0;
};
```

The **color_table** variables indicate which color table files should be used when generating the Wavelet-Stat PostScript output.

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

```
output_prefix = "";
```

This option specifies a string to be used in the output file name. It can be useful for keeping results for different models or variables from overwriting each other.

```
version = "V4.0";
```

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.

7.3.3 wavelet_stat *output*

wavelet_stat produces output in STAT and, optionally, ASCII and NetCDF and PostScript formats. The ASCII output duplicates the STAT output but has the data organized by line type. While the Wavelet-Stat tool currently only outputs one STAT line type, additional line types may be added in future releases. The output files are written to the default output directory or the directory specified by the **-outdir** command-line option.

The output STAT file is named using the following naming convention:
wavelet_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV.stat where **PREFIX** indicates the user-defined output prefix, **HHMMSS** indicates the forecast lead time, and **YYYYMMDD_HHMMSS** indicates the forecast valid time.

The output ASCII files are named similarly:
wavelet_stat_PREFIX_HHMMSSL_YYYYMMDD_HHMMSSV_TYPE.txt where **TYPE** is **isc** to indicate that this is an intensity-scale line type.

The format of the STAT and ASCII output of the Wavelet-Stat tool is similar to the format of the STAT and ASCII output of the Point-Stat tool. Please refer to the tables in section 4.3.3 (point_stat output) for a description of the common output for STAT files types. The information contained in the STAT and **isc** files are identical. However, for consistency with the STAT files produced by other tools, the STAT file will only have column headers for the first 21 fields. The **isc** file contains all headers. The format of the **isc** line type is explained in the following table.

Table 7-2. *Header information for each file wavelet-stat outputs.*

HEADER		
Column Number	**Header Column Name**	**Description**
21	LINE_TYPE	ISC
22	TOTAL	The number of grid points (forecast locations) used
23	TILE_DIM	The dimensions of the tile
24	TILE_XLL	Horizontal coordinate of the lower left corner of the tile
25	TILE_YLL	Vertical coordinate of the lower left corner of the tile
26	NSCALE	Total number of scales used in decomposition
27	ISCALE	The scale at which all information following applies
28	MSE	Mean squared error for this scale
29	ISC	The intensity scale skill score

HEADER		
Column Number	**Header Column Name**	**Description**
30	FENERGY2	Forecast energy squared for this scale
31	OENERGY2	Observed energy squared for this scale
32	BASER	The base rate (not scale dependent)
33	FBIAS	The frequency bias

The Wavelet-Stat tool creates a NetCDF output file containing the raw and decomposed values for the forecast, observation, and difference fields for each combination of variable and threshold value.

The dimensions and variables included in the `wavelet_stat` NetCDF files are described in Tables 7-3 and 7-4.

Table 7-3. *NetCDF dimensions for Wavelet-Stat output.*

`mode` *NetCDF OUTPUT FILE DIMENSIONS*	
NetCDF Dimension	**Description**
x	Dimension of the tile which equals 2^n
y	Dimension of the tile which equals 2^n
scale	Dimension for the number of scales. This is set to n+2, where 2^n is the tile dimension. The 2 extra scales are for the binary image and the wavelet averaged over the whole tile.
tile	Dimension for the number of tiles used

Table 7-4. *Variables contained in Wavelet-Stat NetCDF output.*

Wavelet-stat *NetCDF OUTPUT FILE VARIABLES*		
NetCDF Variable	**Dimension**	**Description**
FCST_FIELD_LEVEL_RAW	tile, x, y	Raw values for the forecast field specified by “FIELD_LEVEL”
OBS_FIELD_LEVEL_RAW	tile, x, y	Raw values for the observation field specified by “FIELD_LEVEL”
DIFF_FIELD_LEVEL_RAW	tile, x, y	Raw values for the difference field (f-o) specified by “FIELD_LEVEL”
FCST_FIELD_LEVEL_THRESH	tile, scale, x, y	Wavelet scale-decomposition of the forecast field specified by “FIELD_LEVEL_THRESH”
OBS_FIELD_LEVEL_THRESH	tile, scale, x, y	Wavelet scale-decomposition of the observation field specified by “FIELD_LEVEL_THRESH”
DIFF_FIELD_LEVEL_THRESH	tile, scale, x, y	Wavelet scale-decomposition of the difference field (f-o) specified by “FIELD_LEVEL_THRESH”

Lastly, the Wavelet-Stat tool creates a PostScript plot summarizing the scale-decomposition approach used in the verification. The PostScript plot is generated using internal libraries and does not depend on an external plotting package. The generation of this PostScript output can be disabled using the `-ps` command line option.

The PostScript plot begins with one summary page illustrating the tiling method that was applied to the domain. The remaining pages depict the Intensity-Scale method that was applied. For each combination of field, tile, and threshold, the binary difference field (f-o) is plotted followed by the difference field for each decomposed scale. Underneath each difference plot, the statistics applicable to that scale are listed. Examples of the PostScript plots can be obtained by running the example cases provided with the MET tarball.

Chapter 8 – The Stat-Analysis Tool

8.1 Introduction

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

The Stat-Analysis tool requires STAT output from Point-Stat, Grid-Stat, and/or Wavelet-Stat. See Sections 4.3.3, 5.3.3 or 7.3.3, respectively, for information on the STAT output format of the Point-Stat, Grid-Stat, and Wavelet-Stat tools.

8.2 Scientific and statistical aspects

The Stat-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, output filename, or wavelet decomposition scale; and (iii) computes specific verification indices such as the GO Index[1] and wind direction statistics. Future functionality may include information about time-trends and/or calculations based on climatology (e.g., anomaly correlation). This section summarizes the capabilities of the Stat-Analysis tool and describes how the GO Index, wind direction, summary statistics, and aggregated statistics are computed.

8.2.1 Filter STAT lines

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

8.2.2 Summary statistics for columns

The Stat-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 10^{th}, 25^{th}, 50^{th}, 75^{th}, and 90^{th} 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

[1] The GO Index is a summary measure for NWP models that is used by the Air Force Weather Agency. It

are resampled, and for each replicated sample, the mean and standard deviation are computed.

8.2.3 Aggregated values from multiple STAT lines

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

8.2.4 Aggregate STAT lines and produce aggregated statistics

The Stat-Analysis tool can be used to aggregate multiple STAT 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 8.2.3. However, rather than writing out the aggregated STAT line itself, the relevant statistics generated from that aggregated line are provided in the output. Specifically, if a contingency table line type (FHO, CTC, PCT, or NBRCTC) has been aggregated, a contingency table statistics (CTS, PSTD, or NBRCTS) line type will be written out. If a partial sums line type (SL1L2 or SAL1L2) has been aggregated, a continuous statistics (CNT) line type will be written out. If the matched pair line type (MPR) has been aggregated, the user may choose the line type to be output (FHO, CTC, CTS, CNT, SL1L2, SAL1L2, PCT, PSTD, PJC, or PRC). Only wind vector statistics are produced from 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 statistics are computed. The confidence intervals are derived from the statistics computed for each replicated sample.

8.2.5 Skill Score Index, including GO Index

The Stat-Analysis tool can be used to calculate the skill score indices by weighting scores for different meteorological fields at different pressure levels and for different lead times. Specifically, the GO Index can be computed. The GO index 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 8-1. The partial sums (SL1L2 lines in the STAT output) for each of these variables at each level and lead time must have

been computed in a previous step. The STAT analysis tool then uses the weights in Table 8-1 to compute values for the GO Index. For a general skill score index, the user can specify the weights and variables to use in the calculations.

Table 8-1. Variables, levels, and weights used to compute the GO Index.

Variable	Level	Weights by lead time			
		12 h	24 h	36 h	48 h
Wind speed	250 hPa	4	3	2	1
	400 hPa	4	3	2	1
	850 hPa	4	3	2	1
	Surface	8	6	4	2
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

8.2.6 Wind Direction Statistics

The Stat-Analysis tool can be used to calculate the error statistics for the wind direction. The vector partial sums (VL1L2) for the UWND and VWND must have been computed in a previous step, i.e. by Point-Stat or Grid-Stat tools. This job computes an average forecast wind direction and an average observed wind direction along with their difference. The output is in degrees. In Point-Stat and Grid-Stat, the UWND and VWND can be verified using thresholds on their values or on the calculated wind speed. If thresholds have been applied, the wind direction statistics are calculated for each threshold.

The first step in verifying wind direction is running the Grid-Stat and/or Point-Stat tools to verify each forecast of interest and generate the VL1L2 line(s). When running these tools, please note:

1. To generate VL1L2 lines, the user must request the verification of the U-component of the wind followed by the V-component, both at the same vertical level.

2. To generate VL1L2 lines, the user must set the “output_flag” to indicate that the VL1L2 line should be computed and written out.

3. The user may select one or more spatial verification regions over which to accumulate the vector partial sums.

4. The user may select one or more wind speed thresholds to be applied to the U and V wind components when computing the VL1L2 lines. It may be useful to investigate the performance of wind forecasts using multiple wind speed thresholds.

Once the VL1L2 lines have been generated for each verification time of interest, the user may run the STAT-Analysis tool to analyze them. The STAT-Analysis job "aggregate_stat", along with the "-output_line_type WDIR" option, reads all of the input VL1L2 lines and computes statistics about the wind direction. When running this job the user is encouraged to use the many STAT-Analysis options to filter the input VL1L2 lines down to the set of lines of interest. The output of the wind direction analysis job consists of two lines with wind direction statistics computed in two slightly different ways. The two output lines begin with "ROW_MEAN_WDIR" and "AGGR_WDIR", and the computations are described below:

1. For the "ROW_MEAN_WDIR" line, each of the input VL1L2 lines is treated separately and given equal weight. The mean forecast wind direction, mean observation wind direction, and the associated error are computed for each of these lines. Then the means are computed across all of these forecast wind directions, observation wind directions, and their errors.

2. For the "AGGR_WDIR" line, the input VL1L2 lines are first aggregated into a single line of partial sums where the weight for each line is determined by the number of points it represents. From this aggregated line, the mean forecast wind direction, observation wind direction, and the associated error are computed and written out.

8.3 Practical information

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

8.3.1 `stat_analysis` *usage*

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

```
Usage: stat_analysis
        -lookin path
        [-out name]
        [-tmp_dir path]
        [-log file]
        [-v level]
        [-config config_file] | [JOB COMMAND LINE]
```

`Stat_analysis` has two required arguments and accepts optional ones.

In the usage statement for the STAT analysis tool, some additional terminology is introduced. In the STAT analysis tool, the term "job" refers to a set of tasks to be performed after applying user-specified options (i.e., "filters"). The filters are used to pare down a collection of output from `grid_stat`, `point_stat`, or `wavelet_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 8-2 and the filtering options are described in section 8.3.2.

Required arguments for `stat_analysis`

1. The `-lookin` **path** specifies the name of a specific STAT file (any file ending in .stat) or the name of a directory where the Stat-Analysis tool will search for STAT files. This option may be used multiple times to specify multiple locations.
2. 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 8.3.2,

Optional arguments for `stat_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 **–**`tmp_dir` **path** option selects the directory for writing out temporary files.
4. The **-**`log` **file** option directs output and errors to the specified log file. All messages will be written to that file as well as cout and cerr. Thus, users can save the messages without having to redirect the output on the command line. The default behavior is no logfile.
5. The `-v` **level** indicates the desired level of verbosity. The contents of "**level**" will override the default setting of 2. Setting the verbosity to 0 will make the tool run with no log messages, while increasing the verbosity above 1 will increase the amount of logging.

An example of the `Stat_analysis` calling sequence is shown below.

```
stat_analysis  -lookin ../out/point_stat
               -config STATAnalysisConfig
```

In this example, the stat analysis tool will search for valid STAT lines located in the ../out/point_stat directory that meet the options specified in the configuration file, config/STATAnalysisConfig.

8.3.2 stat_analysis configuration file

The default configuration file for the stat analysis tool named STATAnalysisConfig_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 STAT 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 STAT analysis configuration file will be retained.

The configuration file for the STAT 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 tool.

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

The user may specify a comma-separated list of model names to be used for all analyses performed. The names must be in double quotation marks. If multiple models are listed, the analyses will be performed on their union. These selections may be further refined by using the "-model" option within the job command lines.

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

The user may specify a comma-separated list of forecast and observation lead times in HH[MMSS] format to be used for any analyses to be performed. If multiple times are listed, the analyses will be performed on their union. These selections may be further refined by using the "-fcst_lead" and "-obs_lead" options within the job command lines.

```
fcst_valid_beg = "";
fcst_valid_end = "";
fcst_valid_hour = "";

obs_valid_beg  = "";
obs_valid_end  = ""
obs_valid_hour  = "";
```

The user may specify the beginning, ending, and instantaneous valid times in YYYYMMDD[_HH[MMSS]] format to be used for all analyses performed. If multiple valid times fall within the valid time window, the analyses will be performed on their union. These selections may be further refined by using the "-fcst_valid_beg", "–fcst_valid_end", "-obs_valid_beg", and "–obs_valid_end" options within the job command line.

```
fcst_init_beg = "";
fcst_init_end = "";
fcst_init_hour = "";

obs_init_beg  = "";
obs_init_end  = "";
obs_init_hour = "";
```

The user may specify the beginning, ending, or exact model initialization times in YYYYMMDD[_HH[MMSS]] format to be used for all analyses performed. If multiple init times fall within the init time window, the analyses will be performed on their union. These selections may be further refined by using the "-fcst_init_beg", "–fcst_init_end", "-obs_init_beg", and "–obs_init_end" options within the job command line.

```
fcst_var[] = [];
obs_var[]  = [];
```

The user may specify a comma-separated list of forecast and observation variable types to be used for any analyses to be performed. If multiple variable types are listed, the

analyses will be performed on their union. These selections may be further refined by using the "-fcst_var" and "-obs_var" options within the job command lines.

---

fcst_lev[] = [];
obs_lev[] = [];

The user may specify a comma-separated list of forecast and observation level types to be used for any analyses to be performed. If multiple level types are listed, the analyses will be performed on their union. These selections may be further refined by using the "-fcst_lev" and "-obs_lev" options within the job command lines.

---

obtype[] = [];

The user may specify a comma-separated list of observation types to be used for all analyses. If multiple observation types are listed, the analyses will be performed on their union. These selections may be further refined by using the "-obtype" option within the job command line.

---

vx_mask[] = [ ];

The user may specify a comma-separated list of verification masking regions to be used for all analyses. If multiple verification masking regions are listed, the analyses will be performed on their union. These selections may be further refined by using the "-vx_mask" option within the job command line.

---

interp_mthd[] = [];

The user may specify a comma-separated list of interpolation methods to be used for all analyses. If multiple interpolation methods are listed, the analyses will be performed on their union. These selections may be further refined by using the "-interp_mthd" option within the job command line.

---

interp_pnts[] = [];

The user may specify a comma-separated list of interpolation points to be used for all analyses. If multiple interpolation points are listed, the analyses will be performed on their union. These selections may be further refined by using the "-interp_pnts" option within the job command line.

```
fcst_thresh[] = [];
obs_thresh[]  = [];
cov_thresh[]  = [];
```

The user may specify comma-separated lists of forecast, observation, and coverage thresholds to be used for any analyses to be performed. If multiple thresholds are listed, the analyses will be performed on their union. These selections may be further refined by using the "-fcst_thresh", "-obs_thresh", and "-cov_thresh" options within the job command lines.

```
alpha[] = [];
```

The user may specify a comma-separated list alpha confidence values to be used for all analyses. If alpha values are listed, the analyses will be performed on their union. These selections may be further refined by using the "-alpha" option within the job command line.

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

The user may specify a comma-separated list of line types to be used for all analyses. If multiple line types are listed, the analyses will be performed on their union. These selections may be further refined by using the "-line_type" option within the job command line.

```
column = [];
weight = [];
```

The column and weight fields are used to define a skill score index. The computation of a single value will be computed from each column and weight value specified. The GO Index is a specific example of a skill score index.

```
jobs[] = [
"-job filter -dump_row ./filter_job.stat"
];
```

The user may specify one or more analysis jobs to be performed on the STAT lines that remain after applying the filtering parameters listed above. Each entry in the joblist contains the task and additional filtering options for a single analysis to be performed. The format for an analysis job is as follows:

-job job_name REQUIRED and OPTIONAL ARGUMENTS

All possible tasks for `job_name` are listed in Table 8-2.

Table 8-2. Description of components of the job command lines for the STAT analysis tool.

Job Name	Description	Required Arguments
filter	Filters out the statistics lines based on applying options	`-dump_row`
summary	Computes the mean, standard deviation, and percentiles (min, 10th, 25th, 50th, 75th, 90th, and max)	`-line_type` `-column`
aggregate	Aggregates the statistics output, computing the statistic specified for the entire collection of valid lines	`-line_type`
aggregate_stat	Aggregates the statistics output, and converts the input line type to the output line type specified	`-line_type` **`-out_line_type`**
go_index	Calculates the GO Index as described in section 8.1.1.	`-fcst_init_begin` `-fcst_init_end`

```
out_alpha = 0.05;

boot = {
   interval = PCTILE;
   rep_prop = 1.0;
   n_rep    = 1000;
   rng      = "mt19937";
   seed     = "";
};
```

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

The **`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.

The **n_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.

The **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.

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

```
vif_flag = 0;
```

The variance inflation factor (VIF) flag indicates whether to apply a first order variance inflation when calculating normal confidence intervals for an aggregated time series of contingency table counts or partial sums. The VIF adjusts the variance estimate for the lower effective sample size caused by autocorrelation of the statistics through time. A value of (0) will not compute confidence intervals using the VIF. A value of (1) will include the VIF, resulting in a slightly wider normal confidence interval.

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

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

```
version = "V4.0";
```

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

8.3.3 Stat-Analysis tool output

The output generated by the Stat-Analysis tool contains statistics produced by the analysis. It also records information about the analysis job that produced the output for each line. Generally, the output is printed to the screen. However, it can be redirected to an output file using the "`-out`" option. The format of output from each STAT job command is described below.

Job: filter

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

The output of this job is the same STAT format described in sections 4.3.3, 5.3.3, and 7.3.3.

Job: summary

This job produces summary statistics for the column name and line type specified by the "`-column`" and "`-line_type`" options. The output of this job type consists of three lines. The first line contains "`JOB_LIST`", followed by a colon, then the filtering and job definition parameters used for this job. The second line contains "`COL_NAME`", followed by a colon, then the column names for the data in the next line. The third line contains the word "`SUMMARY`", followed by a colon, then the total, mean with confidence intervals, standard deviation with confidence intervals, minimum value, percentiles (10^{th}, 25^{th}, 50^{th}, 75^{th}, and 90^{th}) and the maximum value. The output columns are shown in Table 8-3 below.

Table 8-3. Columnar output of "`summary`" job output from the Stat-Analysis tool.

Column Number	Description
1	`SUMMARY:` (job type)
2	Total
3-7	Mean including normal and bootstrap upper and lower confidence limits
8-10	Standard deviation including

Column Number	Description
	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

Job: aggregate

This job aggregates output from the STAT line type specified using the "`-line_type`" argument. The output of this job type is in the same format as the line type specified (see Sections 4.3.3, 5.3.3, and 7.3.3). Again the output consists of three lines. The first line contains "`JOB_LIST`", as described above. The second line contains "`COL_NAME`", followed by a colon, then the column names for the line type selected. The third line contains the name of the line type selected followed by the statistics for that line type.

Job: aggregate_stat

This job is similar to the "`aggregate`" job listed above, however the format of its output is determined by the "`-out_line_type`" argument. Again the output consists of three lines for "`JOB_LIST`", "`COL_NAME`", and the name of the output STAT line, as described above. Valid combinations of the "`-line_type`" and "`-out_line_type`" arguments are listed in Table 8-4 below.

Table 8-4. Valid combinations of "`-line_type`" and "`-out_line_type`" arguments for the "`aggregate_stat`" job.

Input Line Type	Output Line Type
FHO or CTC	CTS
MCTC	MCTS
SL1L2 or SAL1L2	CNT
VL1L2 or VAL1L2	WDIR (wind direction)
PCT	PSTD, PJC, PRC
NBRCTC	NBRCTS
ORANK	RHIST
MPR	CNT, SL1L2, or SAL1L2
MPR	FHO, CTC, CTS, MCTC, MCTS, PCT, PSTD, PJC, or PRC (must specify "`-out_fcst_thresh`" and "-out_obs_thresh" arguments)

Job: go_index

The output from this job consists of three lines, the first two of which contain "JOB_LIST" and "COL_NAME", as described above. The third line contains "GO_INDEX" followed by a colon and then the value computed for the GO Index.

Chapter 9 – The MODE-Analysis Tool

9.1 Introduction

MODE output files can be quite large; currently, these files contain 51 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.

9.2 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 the 10th, 25th, 50th, 75th and 90th percentiles. In addition, the user may specify a "dump" file: the individual MODE lines used to produce the statistics will be written to this file. This option provides the user with a filtering capability. The dump file will consist only of lines that match the specified criteria.

The other option for operating the analysis tool is "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 cluster objects. A dump file may also be specified in this mode.

9.3 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.

9.3.1 **mode_analysis** *usage*

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

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

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 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` / `-cluster`

This toggle indicates whether simple object or cluster object lines should be used.

`-matched` / `-unmatched`

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

Multiple-set string options

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

`-model` value

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

`-fcst_thr` value
`-obs_thr` value

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

`-fcst_var` value
`-obs_var` value

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

`-fcst_lev` value
`-obs_lev` value

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

Multiple-set integer options

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

`-fcst_lead` value
`-obs_lead` value

These options are integers of the form HH[MMSS] specifying an (hour-minute-second) lead time.

`-fcst_init` value
`-obs_init` value

These options are integers of the form HH[MMSS] specifying an (hour-minute-second) model initialization time of day.

`-fcst_accum` value
`-obs_accum` value

These options are integers of the form HHMMSS specifying an (hour-minute-second) accumulation time.

`-fcst_rad` value
`-obs_rad` value

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

Integer max/min options

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

`-area_min` value
`-area_max` value

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

`-area_filter_min` value
`-area_filter_max` value

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

`-area_thresh_min` value
`-area_thresh_max` value

These options are used to indicate minimum/maximum values accepted for the area thresh. The area thresh refers to the number of values of the raw data found within the object that meet the object definition threshold criteria used.

`-intersection_area_min` value
`-intersection_area_max` value

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

`-union_area_min` value
`-union_area_max` value

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

-symmetric_diff_min value
-symmetric_diff_max value

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

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

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

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

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

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

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 other MET tools.

-help

This option prints the usage message.

9.3.2 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.

9.3.3 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 10 – Scripting

10.1 Example scripts for running MET tools

This section provides examples of the use of the Bourne shell, `she`, 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 10-1

```
 1   #!/bin/csh
 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           00000000)0000 $pcp_accum_period \
17      200608$ds_0000 $valid_accum_period \
18         200608$ds.pcp.nc
19           -pcpdir /home/user/my_pcp_dir
20         day=$((day + 1 ))
21 done
```

The first line in Script 10-1 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 precipitation 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 21. Since `pcp_combine` needs dates and times formatted in a certain way, the variable `ads` (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 15–18. We've split the command over several lines so that the script would fit on this page. Line 16 has the `pcp_init_time` argument in this case set to zeroes so that `pcp_combine` will look at all files in the directory. It also has the precipitation accumulation period, as defined in line 3. Line 17 has the valid time, which uses the `ads` variable calculated in lines 9–14. It also contains the valid accumulation period variable, defined in line 4. Line 18 is our output file name. In this example, the output file names include the calendar date and a suffix of "`.pcp.nc`". Line 19 tells `pcp_combine` where to look for input files. Finally, in line 20, 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 observation files are in `/d1/user/obs` and the forecast files are in `/d1/user/fcst`. We want the STAT files written to `/d1/user/gridstat_out`.

Script 10-2

```
 1   #!/bin/csh
 2
 3
 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         $config_file \
21         -outdir /d1/user/gridstat_out \
22         -v2
23      day=$((day + 1 ))
24      done
```

In lines 3–6 of Script 10-2, we define some useful variables: 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 10-1. 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, observation, and config files are named on lines 18 through 20. Note that we use the variable `ads` here. Line 21 names our desired output directory, and line 22 tells `grid_stat` which verbosity level to use. As in Script 10-1, the last line in the loop body (here, line 23) increments `day`.

10.2 Example scripts for use with MODE output files

In script 10-3 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 10-3

```
   1   #!/bin/csh
   2
   3   # Loop through 10 convolution radii.
   4   foreach conv_radius (00 04 08 12 16 24 32 48 64 96)
   5
   6      # Set environmental variable conv_radius so it
   7      # may be used in the mode parameter file.
   8      setenv conv_radius $conv_radius
   9
  10      # Set output directory.
  11      set dir = mode_output/${conv_radius}km
  12      # If output directory does not exist, create it.
  13      if (! -d $dir) mkdir $dir
  14
  15      # Loop through 7 intensity thresholds.
  16      foreach conv_thresh (gt00. ge01. ge02. ge03. ge04.
ge05. ge06.)
  17
  18         # Set environmental variable conv_thresh so it
  19         # may be used in the mode parameter file.
  20         setenv conv_thresh \"$conv_thresh\"
  21
  22         # Set output directory.
```

```
  23         set dir =
mode_output/${conv_radius}km/${conv_thresh}
  24         # Create directory if it does not exist.
  25         if (! -d $dir) mkdir $dir
  26
  27         # execute mode with output directory $dir
  28         mode pcp/wrf2caps/wrf2caps_2005053100.g240.f24.nc \
  29            pcp/st2/ST2ml_2005060100.g240.nc \
  30            mode_output/WrfModeConfig_default \
  31            -outdir $dir
  32
  33      end
  34   end
```

Line 1 is similar to Line 1 of scripts 10-1 and 10-2. 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_thresh` and `obs_conv_thresh` (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 script10-3.csh).

excerpt from mode parameter file to which script 10-3 refers

```
   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_thresh   = 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
   131    //
   132    fcst_conv_thresh  = ${conv_thresh};
   133    obs_conv_thresh   = ${conv_thresh};
```

Chapter 11 – 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/) as examples to assist users.

11.1 Grid-Stat tool examples

The plots in Figure 11-1 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.

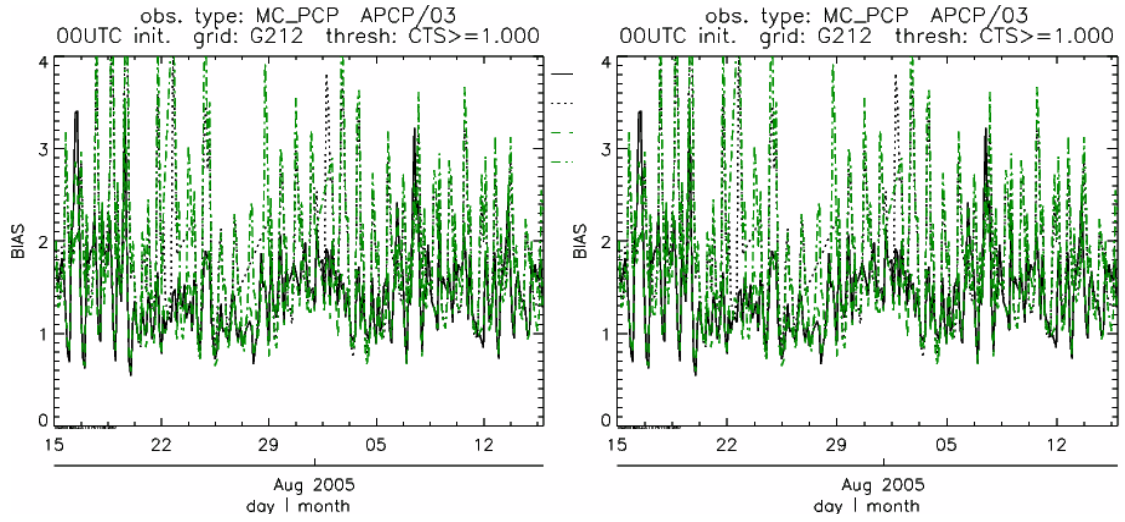


***Figure 11-1**: Time series of forecast bias and Gilbert Skill Score for several numerical models (differentiated by line-type and color) over a 32-day period in July/Aug 2005. Scores are based on the forecast of 1 mm or greater precipitation at 3-h intervals. Models were run for 24 h and were initiated every day at 00 UTC.*

A similar plot is shown in Fig. 11-2, 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.

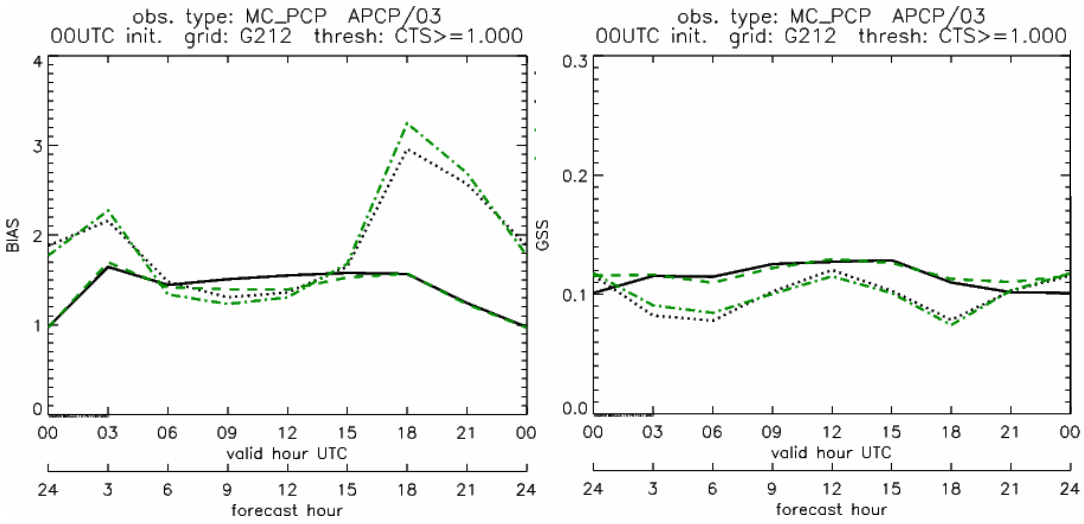


***Figure 11-2**: Time series of forecast area bias and Gilbert Skill Score for four model configurations (different lines) stratified by time-of-day. The data used to create these figures were the same as used for Fig. 10-1.*

11.2 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 11-3 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.

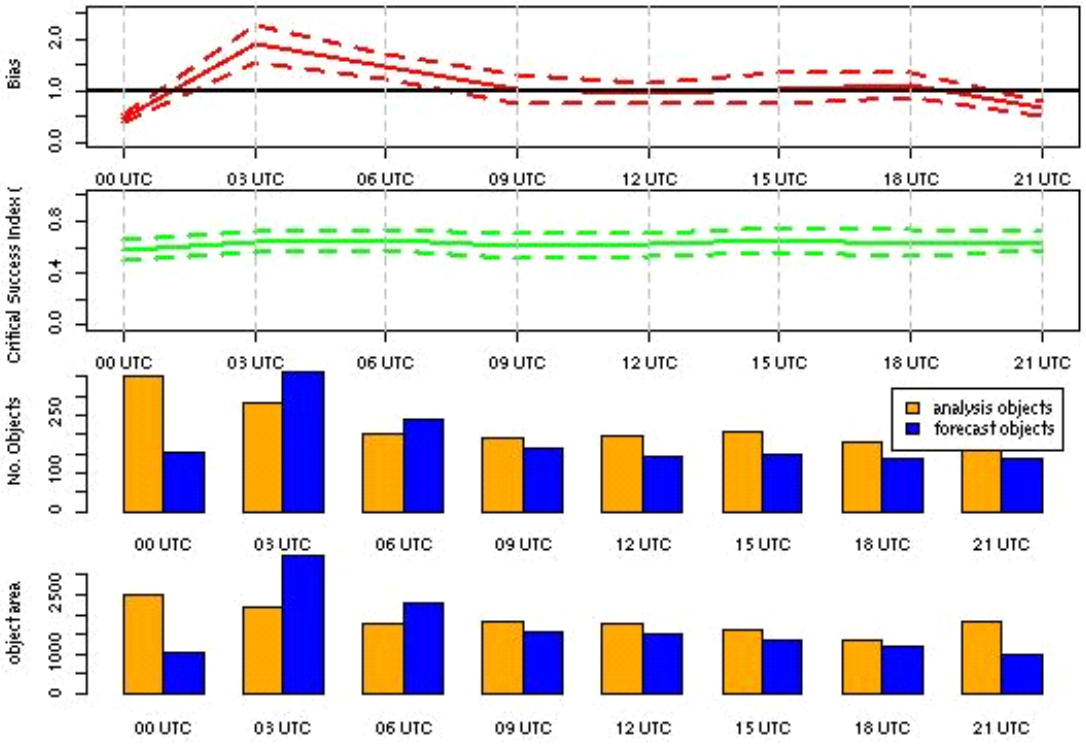


***Figure 11-3**: Traditional verification scores applied to output of the MODE tool, computed by defining matched observed objects to be hits, unmatched observed objects to be misses, and unmatched forecast objects to be false alarms; weighted by object area. Bar plots show numbers (pennultimate row) 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. 11-4, 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. 11-4 (top diagram) about 25% of the time the forecast object centroids are west of the observed object centroids, whereas in Fig. 11-4 (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 11-4 (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.

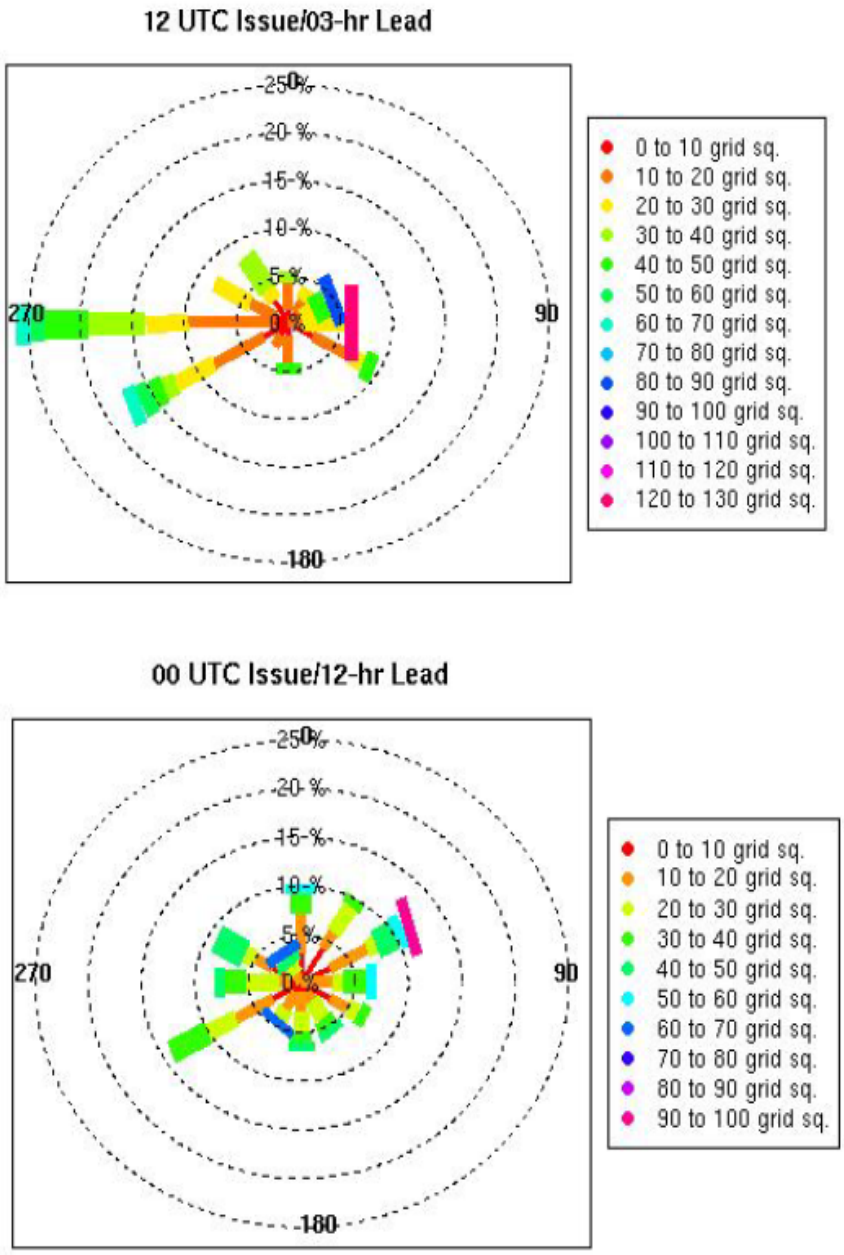


***Figure 11-4**: Circle histograms showing object centroid angles and distances (see text for explanation).*

References

Bradley, A.A., S.S. Schwartz, and T. Hashino, 2008: Sampling Uncertainty and Confidence Intervals for the Brier Score and Brier Skill Score. *Weather and Forecasting*, **23**, 992–1006.

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.

Casati, B., Ross, G., and Stephenson, D. 2004: A new intensity-scale approach for the verification of spatial precipitation forecasts. *Meteorol. Appl*. **11**, 141-154

Davis, C.A., B.G. Brown, and R.G. Bullock, 2006a: Object-based verification of precipitation forecasts, Part I: Methodology and application to mesoscale rain areas. *Monthly Weather Review*, **134**, 1772-1784.

Davis, C.A., B.G. Brown, and R.G. Bullock, 2006b: Object-based verification of precipitation forecasts, Part II: Application to convective rain systems. *Monthly Weather Review*, **134**, 1785-1795.

Dawid, A. P., 1984: Statistical theory: The prequential approach. *J. Roy. Stat. Soc*, **A147**:278–292.

Ebert, E.E., 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.

Gneiting, T., Westveld, A., Raferty, A. and Goldman, T, 2004: *Calibrated Probabilistic Forecasting Using Ensemble Model Output Statistics and Minimum CRPS Estimation.* Technical Report no. 449, Department of Statistics, University of Washington. [Available online at http://www.stat.washington.edu/www/research/reports/]

Hamill, T. M., 2001: Interpretation of rank histograms for verifying ensemble forecasts. *Mon. Wea. Rev.*, **129**, 550–560.

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 is PrepBufr used?
A. The first goal of MET was to replicate the capabilities of existing verification packages and make these capabilities available to both the DTC and the public.

Q. Why is GRIB used?
A. Forecast data from both WRF cores can be processed into GRIB format, and it is a commonly accepted output format for many NWP models.

Q. Is GRIB2 supported?
A. Yes, new for this release, forecast output in GRIB2 format can be read by MET. Be sure to compile the GRIB2 code by setting the appropriate flags in the user_defs.mk file.

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 MET tools are available by simply typing the name of the executable in MET's `bin/` directory. Example scripts available in the MET's `scripts/` directory show examples of how one might use these commands on example datasets. Here are suggestions on other things to check if you are having problems installing or running MET.

MET won't compile

- Are you using the correct version of the Makefile (Makefile_gnu for using the GNU compilers; Makefile_pgi for the PGI compilers, Makefile_intel for Intel 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 Scientific 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?

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)
- Polar Stereographic Projection (Southern)
- Mercator Projection
- Lat/Lon Projection

B.2 Grids

All of NCEP's pre-defined grids that reside on one of the projections listed above are implemented in MET. The user may specify one of these NCEP grids in the configuration files as "GNNN" where NNN is the 3-digit NCEP grid number. Defining a new masking grid in MET would involve modifying the vx_data_grids library and recompiling.

Please see NCEP's website for a description and plot of these pre-defined grids:
http://www.nco.ncep.noaa.gov/pmb/docs/on388/tableb.htmln

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 polylines are listed below:

- APL.poly for the Appalachians
- ATC.poly for the Arctic Region
- CAM.poly for Central America
- CAR.poly for the Caribbean Sea
- ECA.poly for Eastern Canada
- GLF.poly for the Gulf of Mexico
- GMC.poly for the Gulf of Mexico Coast
- GRB.poly for the Great Basin
- HWI.poly for Hawaii
- LMV.poly for the Lower Mississippi Valley
- MDW.poly for the Midwest
- MEX.poly for Mexico
- NAK.poly for Northern Alaska
- NAO.poly for Northern Atlantic Ocean
- NEC.poly for the Northern East Coast
- NMT.poly for the Northern Mountain Region
- NPL.poly for the Northern Plains
- NPO.poly for the Northern Pacific Ocean
- NSA.poly for Northern South America
- NWC.poly for Northern West Coast
- PRI.poly for Puerto Rico and Islands
- SAK.poly for Southern Alaska
- SAO.poly for the Southern Atlantic Ocean
- SEC.poly for the Southern East Coast
- SMT.poly for the Southern Mountain Region
- SPL.poly for the Southern Plains
- SPO.poly for the Southern Pacific Ocean
- SWC.poly for the Southern West Coast
- SWD.poly for the Southwest Desert
- WCA.poly for Western Canada
- EAST.poly for the Eastern United States (consisting of APL, GMC, LMV, MDW, NEC, and SEC)
- WEST.poly for the Western United States (consisting of GRB, NMT, NPL, NWC, SMT, SPL, SWC, and SWD)
- CONUS.poly for the Continental United States (consisting of EAST and WEST)

Appendix C – Verification Measures

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

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

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

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

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 "BASER" in CTS output (Table 4-5)

The base rate is defined as $\bar{o} = \frac{n_{11} + n_{01}}{T} = \frac{n_{\cdot 1}}{T}$. 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 "FMEAN" in CTS output (Table 4-5)

The mean forecast value is defined as $\bar{f} = \frac{n_{11} + n_{10}}{T} = \frac{n_{1\cdot}}{T}$. 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 4-5)

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 4-5)

Frequency Bias is the ratio of the total number of forecasts of an event to the total number of observations of the event. It is defined as $\text{Bias} = \frac{n_{11} + n_{10}}{n_{11} + n_{01}} = \frac{n_{1.}}{n_1}$. A "good" value of Frequency Bias is close to 1; a value greater than 1 indicates the event was forecasted too frequently and a value less than 1 indicates the event was not forecasted frequently enough.

Probability of Detection (POD)
Called "H_RATE" in FHO output (Table 4-3);
Called "PODY" in CTS output (Table 4-5)

POD is defined as $\text{POD} = \frac{n_{11}}{n_{11} + n_{01}} = \frac{n_{11}}{n_1}$. It is the fraction of events that were correctly forecasted to occur. POD is equivalent to the H value computed by the NCEP verification system and is also known as the **hit rate**. POD ranges from 0 to 1; a perfect forecast would have POD = 1.

Probability of False Detection (POFD)
Called "POFD" in CTS output (Table 4-5)

POFD is defined as $\text{POFD} = \frac{n_{10}}{n_{10} + n_{00}} = \frac{n_{10}}{n_{.0}}$. It is the proportion of non-events that were forecast to be events. POFD is also often called the **False Alarm Rate**[2]. POFD ranges from 0 to 1; a perfect forecast would have POFD = 0.

Probability of Detection of the non-event (PODn)
Called "PODN" in CTS output (Table 4-5)

PODn is defined as $\text{PODn} = \frac{n_{00}}{n_{10} + n_{00}} = \frac{n_{00}}{n_{.0}}$. It is the proportion of non-events that were correctly forecasted to be non-events. Note that PODn = 1 – POFD. PODn ranges from 0 to 1. Like POD, a perfect forecast would have PODn = 1.

False Alarm Ratio (FAR)
Called "FAR" in CTS output (Table 4-5)

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 the event did not occur. FAR ranges from 0 to 1; a perfect forecast would have FAR = 0.

---

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

Critical Success Index (CSI)
Called "CSI" in CTS output (Table 4-5)

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

Gilbert Skill Score (GSS)
Called "GSS" in CTS output (Table 4-5)

GSS is based on the CSI, corrected for the number of hits that would be expected by chance. In particular, $\mathrm{GSS} = \frac{n_{11} - C_1}{n_{11} + n_{10} + n_{01} - C_1}$, 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 4-5)

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

Heidke Skill Score (HSS)
Called "HSS" in CTS output (Table 4-5)

HSS is a skill score based on Accuracy, where the Accuracy is corrected by the number of correct forecasts that would be expected by chance. In particular, $\mathrm{HSS} = \frac{n_{11} + n_{00} - C_2}{T - C_2}$, where $C_2 = \frac{(n_{11} + n_{10})(n_{11} + n_{01}) + (n_{01} + n_{00})(n_{10} + n_{00})}{T}$. HSS can range from minus infinity to 1. A perfect forecast would have HSS = 1.

Odds Ratio (OR)
Called "ODDS" in CTS output (Table 4-5)

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

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.

Mean forecast
Called "FBAR" in CNT output (Table 4-6)
Called "FBAR" in SL1L2 output (Table 4-11)

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

Mean observation
Called "OBAR" in CNT output (Table 4-6)
Called "FBAR" in SL1L2 output (Table 4-11)

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

Forecast standard deviation
Called "FSTDEV" in CNT output (Table 4-6)

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

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

Observation standard deviation
Called "OSTDEV" in CNT output (Table 4-6)

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

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

Pearson Correlation Coefficient
Called "PR_CORR" in CNT output (Table 4-6)

The **Pearson correlation coefficient**, ***r***, measures the strength of linear association between the forecasts and observations. The Pearson correlation coefficient is defined as $r = \frac{\sum_{i=1}^{T}(f_i - \bar{f})\,(o_i - \bar{o})}{\sqrt{\sum\,(f_i - \bar{f})^2}\sqrt{\sum(o_i - \bar{o})^2}}$.

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 4-6)

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

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

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

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

Kendall's tau statistic (τ)
Called "KT_CORR" in CNT output (Table 4-6)

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

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

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 4-6)

The Mean Error, **ME**, is a measure of overall bias for continuous variables; in particular ME = Bias. It is defined as $\mathrm{ME} = \frac{1}{n}\sum_{i=1}^{n}(f_i - o_i) = \overline{f} - \overline{o}$.

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

Multiplicative Bias
Called "MBIAS" in CNT output (Table 4-6)

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

$$\text{MBIAS} = \frac{\overline{f}}{\overline{o}}$$

Mean-squared error (MSE)
Called "MSE" in CNT output (Table 4-6)

MSE measures the average squared error of the forecasts. Specifically, $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(f_i - o_i\right)^2$.

Root-mean-squared error (RMSE)
Called "RMSE" in CNT output (Table 4-6)

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

Standard deviation of the error
Called "ESTDEV" in CNT output (Table 4-6)

Bias-Corrected MSE
Called "BCMSE" in CNT output (Table 4-6)

MSE and **RMSE** are strongly impacted by large errors. They also are strongly impacted by large bias (**ME**) values. **MSE** and **RMSE** can range from 0 to infinity. A perfect forecast would have **MSE** = **RMSE** = 0.

MSE can be re-written as

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

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

Note that the 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 4-6)

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 4-6)

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

Scalar L1 and L2 values
Called "FBAR", "OBAR", "FOBAR", "FFBAR", and "OOBAR" in SL1L2 output (Table 4-11)

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

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

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

$$\mathrm{FOBAR} = Mean(f \cdot o) = \frac{1}{n}\sum_{i=1}^{n} f_i \cdot o_i$$

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

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

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

Scalar anomaly L1L2 values
Called "FABAR", "OABAR", "FOABAR", "FFABAR", "OOABAR" in SAL1L2 output (Table 4-12)

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:

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

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

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

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

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

Vector L1 and L2 values
Called "UFBAR", "VFBAR", "UOBAR", "VOBAR", "UVFOBAR", "UVFFBAR", "UVOOBAR" in VL1L2 output (Table 4-13)

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:

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

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

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

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

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

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

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

Vector anomaly L1 and L2 values
Called "UFABAR", "VFABAR", "UOABAR", "VOABAR", "UVFOABAR", "UVFFABAR", "UVOOABAR" in VAL1L2 output (Table 4-14)

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

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

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

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

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

$$\text{UVFOABAR} = \text{Mean}[(u_f - u_c)(u_o - u_c) + (v_f - v_c)(v_o - v_c)]$$

$$= \frac{1}{n}\sum_{i=1}^{n}[(u_{fi} - u_c)(u_{oi} - u_c) + (v_{fi} - v_c)(v_{oi} - v_c)]$$

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

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

C.3 MET verification measures for probabilistic forecasts

The results of the probabilistic verification methods that are included in the Point-Stat, Grid-Stat, and Stat-Analysis tools are summarized using a variety of measures. MET treats probabilistic forecasts as categorical, divided into bins by user-defined thresholds between zero and one. For the categorical measures, if a forecast probability is specified in a formula, the mid-point value of the bin is used. These measures include the Brier Score (BS) with confidence bounds (Bradley 2008); the joint distribution, calibration-refinement, likelihood-base rate (Wilks 2006); and receiver operating characteristic information. Using these statistics, reliability and discrimination diagrams can be produced.

The verification statistics for probabilistic forecasts of dichotomous variables are formulated using a contingency table such as the one shown in Table C-2. In this table *f* represents the forecasts and *o* represents the observations; the two possible forecast and observation values are represented by the values 0 and 1. The values in Table C-2 are counts of the number of occurrences of all possible combinations of forecasts and observations.

Table C-2*: nx2 contingency table in terms of counts. The 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")**	
p_1 = midpoint of (0 and threshold1)	n_{11}	n_{10}	$n_{1.} = n_{11} + n_{10}$
p_2 = midpoint of (threshold1 and threshold2)	n_{21}	n_{20}	$n_{2.} = n_{21} + n_{20}$
. . .	. . .	. . .	. . .
p_j = midpoint of (threshold*i* and 1)	n_{i1}	n_{i0}	$n_{j} = n_{j1} + n_{j0}$
Total	$n_{.1} = \Sigma n_{i1}$	$n_{.0} = \Sigma n_{i0}$	$T = \Sigma n_i$

Reliability – A component of the Brier score. Reliability measures the average difference between forecast probability and average observed frequency. Ideally, this measure should be zero as larger numbers indicate larger differences. For example, on occasions when rain is forecast with 50% probability, it should actually rain half the time.

$$\text{Reliability}_i = \frac{1}{T}\sum n_i \left(p_i - \overline{o}_i \right)^2$$

Resolution – A component of the Brier score that measures how well forecasts divide events into subsets with different outcomes. Larger values of resolution are best since it is desirable for event frequencies in the subsets to be different than the overall event frequency.

$$\text{Resolution} = \frac{1}{T} n_{i.} (\overline{o}_i - \overline{o})^2$$

Uncertainty – A component of the Brier score. For probabilistic forecasts, uncertainty is a function only of the frequency of the event. It does not depend on the forecasts, thus there is no ideal or better value. Note that Uncertainty is equivalent to the variance of the event occurrence.

$$\text{Uncertainty} = \frac{n_{.1}}{T}\left(1 - \frac{n_{.1}}{T} \right)$$

Brier score

The Brier score is the mean squared *probability* error. In MET, the Brier Score (BS) is calculated from the nx2 contingency table via the equation, $\text{BS} = \frac{1}{T}\left[n_{.1}(p_1 - 1)^2 + n_{.0} p_0^2 \right]$ The equation you will most often see in references uses the individual probability forecasts (p_i) and the corresponding observations (o_i), and is given as $\text{BS} = \frac{1}{T}\sum_i (p_i - o_i)^2$. This equation is equivalent when the midpoints of the binned probability values are used as the p_i .

BS can be partitioned into three terms: (1) reliability, (2) resolution, and (3) uncertainty (Murphy, 1973).

$$\text{BS} = \frac{1}{T}\sum_i (p_i - o_i)^2 = \frac{1}{T}\sum_i \; n_{i.}(p_i - \overline{o}_i)^2 - \frac{1}{T}\sum_i \; n_{i.}(\overline{o}_i - \overline{o})^2 + \overline{o}(1 - \overline{o})$$

This score is sensitive to the base rate or climatological frequency of the event. Forecasts of rare events can have a good BS without having any actual skill. Since Brier score is a measure of error, smaller values are better.

OY_TP – Observed Yes Total Proportion. This is the cell probability for row *i,* column *j*=1 (observed event), a part of the joint distribution (Wilks, 2006). Along with ON_TP, this set of measures provides information about the joint distribution of forecasts and events. There are no ideal or better values.

$$OY\_TP_i = \frac{n_{i1}}{T} = probability(o_{i1})$$

ON_TP – Observed No Total Proportion. This is the cell probability for row *i,* column *j*=0 (observed non-event), a part of the joint distribution (Wilks, 2006). Along with OY_TP, this set of measures provides information about the joint distribution of forecasts and events. There are no ideal or better values.

$$ON\_TP_i = \frac{n_{i0}}{T} = probability(o_{i0})$$

Calibration – Calibration is the conditional probability of an event given each probability forecast category (i.e. each row in the nx2 contingency table). This set of measures is paired with refinement in the calibration-refinement factorization discussed in Wilks (2006). A well-calibrated forecast will have calibration values that are near the forecast probability. For example, a 50% probability of precipitation should ideally have a calibration value of 0.5. If the calibration value is higher, then the probability has been underestimated, and vice versa.

$$calibration_i = \frac{n_{i1}}{n_{1.}} = probability(o_1 \mid p_i)$$

Refinement – The relative frequency associated with each forecast probability, sometimes called the marginal distribution or row probability. This measure ignores the event outcome, and simply provides information about the frequency of forecasts for each probability category. This set of measures is paired with the calibration measures in the calibration-refinement factorization discussed by Wilks (2006).

$$refinement_i = \frac{n_{i.}}{T} = probability(p_i)$$

Likelihood – Likelihood is the conditional probability for each forecast category (row) given an event and a component of the likelihood-base rate factorization; see Wilks (2006) for details. This set of measures considers the distribution of forecasts for only the cases when events occur. Thus, as the forecast probability increases, so should the likelihood. For example, 10% probability of precipitation forecasts should have a much smaller likelihood value than 90% probability of precipitation forecasts.

$$likelihood_i = \frac{n_{i1}}{n_{.1}} = probability(p_i | o_1)$$

Likelihood values are also used to create "discrimination" plots that compare the distribution of forecast values for events to the distribution of forecast values for non-events. These plots show how well the forecasts categorize events and non-events. The distribution of forecast values for non-events can be derived from the POFD values computed by MET for the user-specified thresholds.

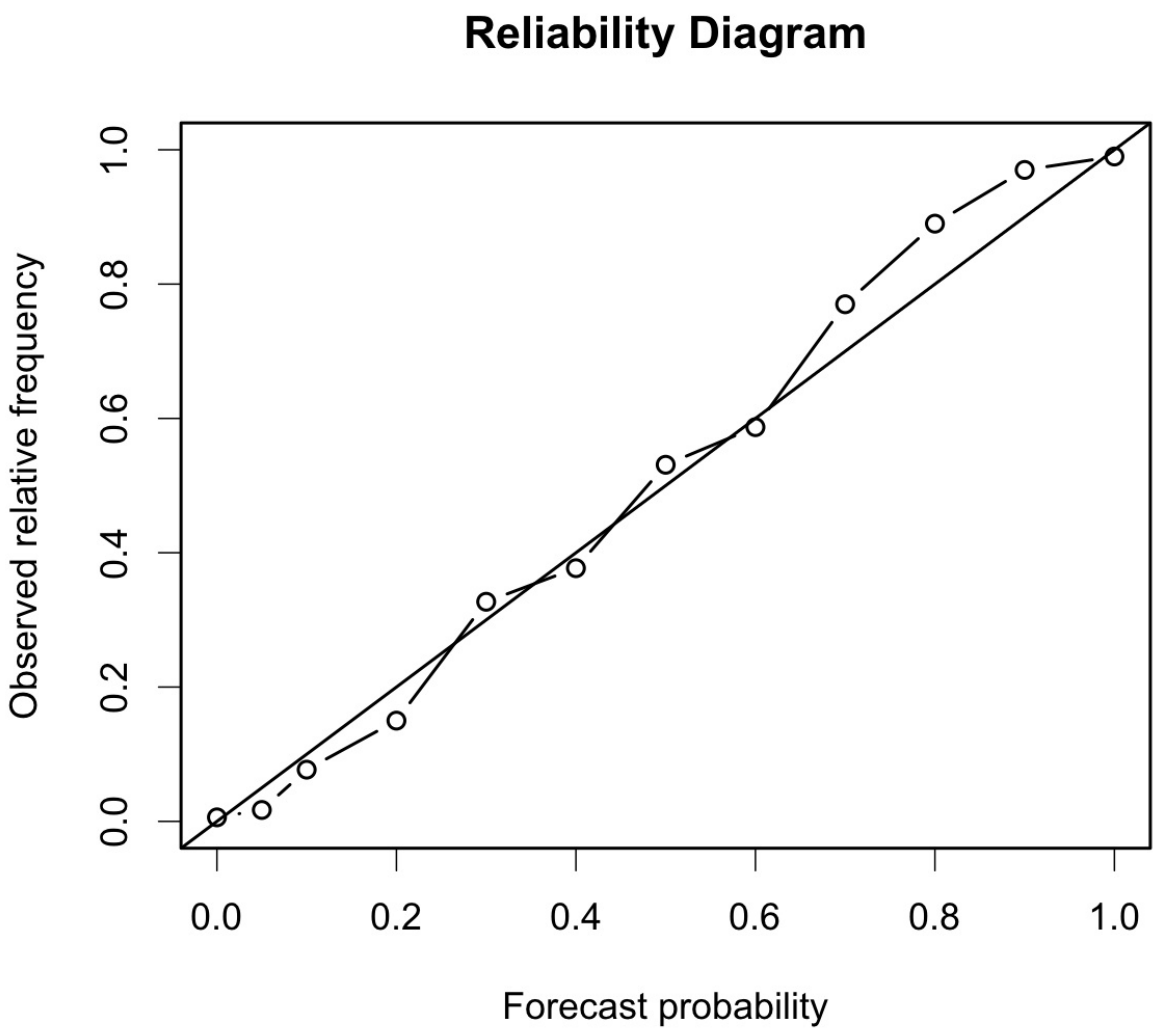
Base Rate – This is the probability of an event for each forecast category p_i (row), i.e. the *conditional* base rate. This set of measures if paired with likelihood in the likelihood-base rage factorization, see Wilks (2006) for further information. This measure is calculated for each row of the contingency table. Ideally, the event should become more frequent as the probability forecast increases.

$$Base\ Rate_i = \frac{n_{i1}}{n_{i.}} = probability(o_{i1})$$

Reliability diagram –

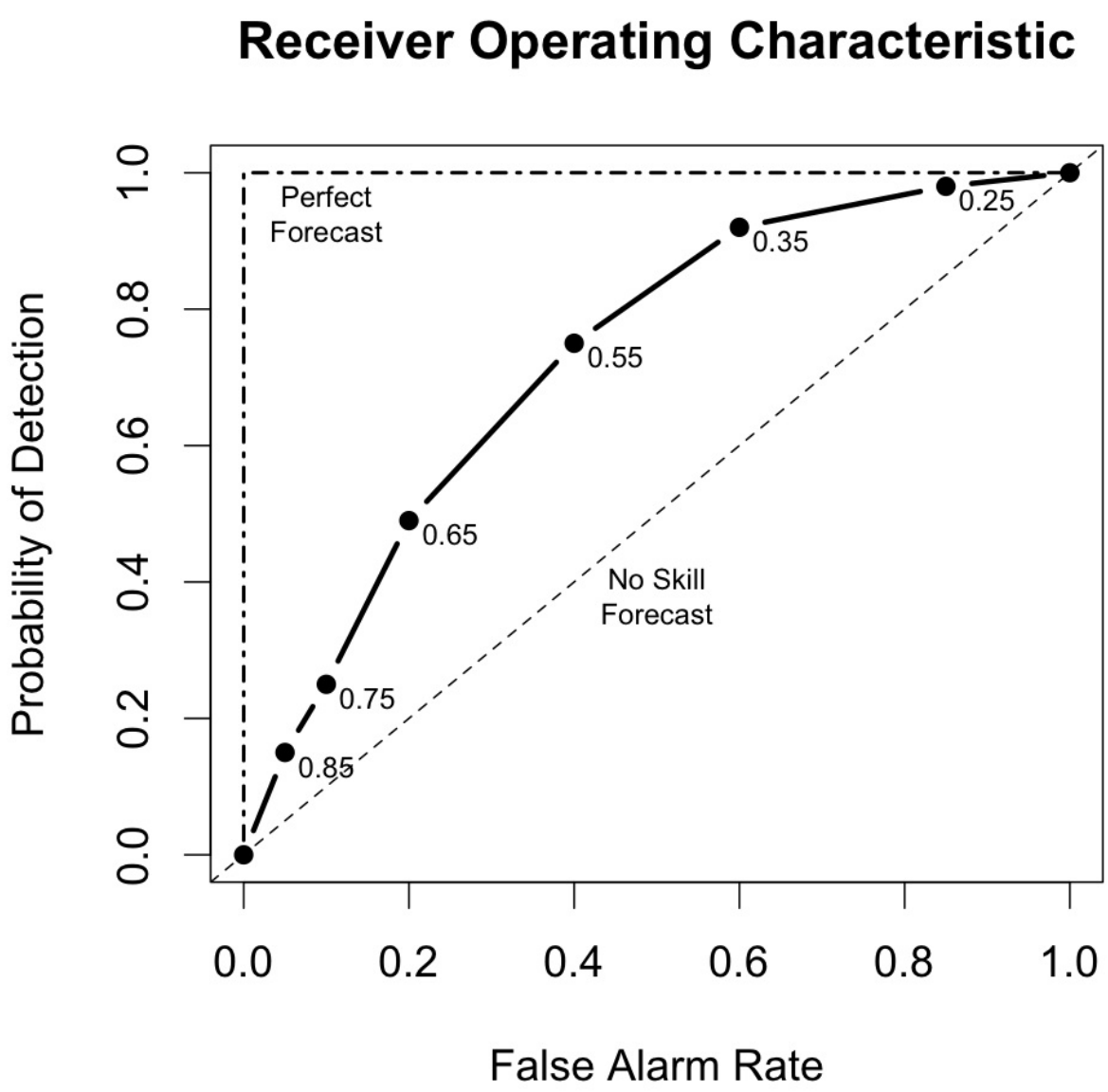
The reliability diagram is a plot of the observed frequency of events versus the forecast probability of those events, with the range of forecast probabilities divided into categories.

The ideal forecast (i.e., one with perfect reliability) has conditional observed probabilities that are equivalent to the forecast probability, on average. On a reliability plot, this equivalence is represented by the one-to-one line (the solid line in the figure below). So, better forecasts are closer to the diagonal line and worse ones are farther away. The distance of each point from the diagonal gives the conditional bias. Points that lie below the diagonal line indicate over-forecasting; in other words, the forecast probabilities are too large. The forecast probabilities are too low when the points lie above the line. The reliability diagram is conditioned on the forecasts so it is often used in combination with the ROC, which is conditioned on the observations, to provide a "complete" representation of the performance of probabilistic forecasts.



Receiver operating characteristic

MET produces hit rate (POD) and false alarm rate (POFD) values for each user-specified threshold. This information can be used to create a scatter plot of POFD vs. POD. When the points are connected, the plot is generally referred to as the receiver operating characteristic (ROC) curve (also called the "relative operating characteristic" curve). See the area under the ROC curve (AUC) entry for related information.



An ROC plot is shown for an example set of forecasts, with a solid line connecting the points for six user-specified thresholds (0.25, 0.35, 0.55, 0.65, 0.75, 0.85). The diagonal dashed line indicates no skill while the dash-dot line shows the ROC for a perfect forecast.

An ROC curve shows how well the forecast discriminates between two outcomes, so it is a measure of resolution. The ROC is invariant to linear transformations of the forecast, and is thus unaffected by bias. An unbiased (i.e., well-calibrated) forecast can have the same ROC as a biased forecast, though most would agree that an unbiased forecast is “better”. Since the ROC is conditioned on the observations, it is often paired with the reliability diagram, which is conditioned on the forecasts.

Area Under the ROC curve (AUC)

The area under the receiver operating characteristic (ROC) curve is often used as a single summary measure. A larger AUC is better. A perfect forecast has AUC=1. Though the minimum value is 0, an AUC of 0.5 indicates no skill.

The area under the curve can be estimated in a variety of ways. In MET, the simplest trapezoid method is used to calculate the area. AUC is calculated from the series of hit rate (POD) and false alarm rate (POFD) values (see the ROC entry below) for each user-specified threshold.

$$AUC = \frac{1}{2} \sum_{i=1}^{Nthresh} (POD_{i+1} + POD_i)(POFD_{i+1} - POFD_i)$$

C.4 MET verification measures for ensemble forecasts

CRPS

The continuous ranked probability score (CRPS) is the integral, over all possible thresholds, of the Brier scores (Gneiting *et al*, 2004). In MET, the CRPS calculation uses a normal distribution fit to the ensemble forecasts. In many cases, use of other distributions would be better.

WARNING: The normal distribution is probably a good fit for temperature and pressure, and possibly a not horrible fit for winds. However, the normal approximation will not work on most precipitation forecasts and may fail for many other atmospheric variables.

Closed form expressions for the CRPS are difficult to define when using data rather than distribution functions. However, if a normal distribution can be assumed, then the following equation gives the CRPS for each individual observation (denoted by a lowercase crps) and the corresponding distribution of forecasts.

$$\text{crps}_i\ (N(\mu,\sigma^2),y) = \sigma\left(\frac{y-\mu}{\sigma}\left(2\Phi\left(\frac{y-\mu}{\sigma}\right)-1\right)+2\varphi\left(\frac{y-\mu}{\sigma}\right)-\frac{1}{\sqrt{\pi}}\right)$$

In this equation, the y represents the event threshold. The estimated mean and standard deviation of the ensemble forecasts (μ and σ) are used as the parameters of the normal distribution. The values of the normal distribution are represented by the probability density function (PDF) denoted by φ and the cumulative distribution function (CDF), denoted in the above equation by Φ.

The overall CRPS is calculated as the average of the individual measures. In equation form.

$$\text{CRPS} = average(crps\ ) = 1/N \sum_{i-1}^{N} crps_i$$

The score can be interpreted as a continuous version of the mean absolute error (MAE). Thus, the score is negatively oriented, so smaller is better. Further, similar to MAE, bias will inflate the CRPS. Thus, bias should also be calculated and considered when judging forecast quality using CRPS.

IGN

The ignorance score (IGN) is the negative logarithm of a predictive probability density function (Gneiting *et al*, 2004). In MET, the IGN is calculated based on a normal approximation to the forecast distribution (i.e. a normal pdf is fit to the forecast values). This approximation may not be valid, especially for discontinuous forecasts like

precipitation, and also for very skewed forecasts. For a single normal distribution *N* with parameters μ and σ, the ignorance score is:

$$ign(N(\mu,\sigma),y) = \frac{1}{2}\ln(2\pi\sigma^2) + \frac{(y-\mu)^2}{\sigma^2}$$

Accumulation of the ignorance score for many forecasts is via the average of individual ignorance scores. This average ignorance score is the value output by the MET software. Like many error statistics, the IGN is negatively oriented, so smaller numbers indicate better forecasts.

PIT

The probability integral transform (PIT) is the analog of the rank histogram for a probability distribution forecast (Dawid, 1984). Its interpretation is the same as that of the verification rank histogram: Calibrated probabilistic forecasts yield PIT histograms that are flat, or uniform. Underdispersed (not enough spread in the ensemble) forecasts have U-shaped PIT histograms while overdispersed forecasts have bell shaped histograms. In MET, the PIT calculation uses a normal distribution fit to the ensemble forecasts. In many cases, use of other distributions would be better.

RANK

The rank of an observation, compared to all members of an ensemble forecast, is a measure of dispersion of the forecasts (Hamill, 2001). When ensemble forecasts possesses the same amount of variability as the corresponding observations, then the rank of the observation will follow a discrete uniform distribution. Thus, a rank histogram will be approximately flat.

The rank histogram does not provide information about the accuracy of ensemble forecasts. Further, examination of "rank" only makes sense for ensembles of a fixed size. Thus, if ensemble members are occasionally unavailable, the rank histogram should not be used. The PIT may be used instead.

C.5 MET verification measures for neighborhood methods

The results of the neighborhood verification approaches that are included in the Grid-Stat tool are summarized using a variety of measures. These measures include the Fractions Skill Score (FSS) and the Fractions Brier Score (FBS). MET also computes

traditional contingency table statistics for each combination of threshold and neighborhood window size.

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

- Base Rate (called "BASER" in 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 $\mathrm{FBS} = \frac{1}{N}\sum_{N}\left[\left\langle P_f \right\rangle_s - \left\langle P_o \right\rangle_s\right]^2$, where N is the number of neighborhoods; $\left\langle P_f \right\rangle_s$ 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 $\left\langle P_o \right\rangle_s$ 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 $\mathrm{FSS} = 1 - \dfrac{\mathrm{FBS}}{\frac{1}{N}\left[\sum_N \langle P_f \rangle_s^2 + \sum_N \langle P_o \rangle_s^2\right]}$, 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

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

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 $V(\bar{X})$, which is obtained here by replacing σ by its estimate, $\hat{\sigma}$, where $\hat{\sigma} = \sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2 \Big/ (n-1)$.

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

$$\mu \pm t_{\alpha/2,\upsilon-1} \cdot \frac{\sigma}{\sqrt{n}}, \tag{D.2}$$

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, $V(\theta)$. 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.

$\hat{\theta}$	$V(\theta)$
Forecast/Observation Mean	$V(\bar{X}) = \frac{\sigma_x}{\sqrt{n}}$ where σ_x emphasizes that this is the estimated standard deviation of the underlying sample.
Mean error	$V(\bar{F} - \bar{O}) = \frac{\sigma_{F-O}}{\sqrt{n}}$, where σ_{F-O} emphasizes that this is the estimated standard deviation of the errors, $F - O$.
Brier Score (BS)	$V(BS) = \frac{1}{T}\left[\sum F^4 + \bar{O}\left(1 - 4\sum F^3_{F\|O=1} + 6\sum F^2_{F\|O=1} - 4\sum F_{F\|O=1}\right) - BS^2\right]$ where F is the *probability* forecast and O is the observation. See Bradley *et al* (2008) for derivation and details.
Peirce Skill Score (PSS)	$V(PSS) = \sqrt{\frac{H(1-H)}{n_H} + \frac{F(1-F)}{n_F}}$, where H is the hit rate, F the false alarm rate, n_H the number of hits and misses, and n_F the number of false alarms and correct negatives.
Logarithm of the odds ratio (OR)	$V(\ln(OR)) = \sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}$, where the values in the denominators are the usual contingency table counts.

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

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

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

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

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

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

respectively, where $\chi^2_{\alpha,\nu}$ is the 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

$$\left(\frac{e^{2c_1}-1}{e^{2c_1}+1}, \frac{e^{2c_u}-1}{e^{2c_u}+1}\right), \tag{D.5}$$

where $c_1 = \nu - \frac{z_{\alpha/2}}{\sqrt{n-3}}$ 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, and the simplest to understand. It is merely the $\alpha/2$ and $1-\alpha/2$ percentiles from the sample of statistics. Unfortunately, however, it has been shown that this interval is too optimistic in practice (i.e., it doesn't have accurate coverage). One solution is to use the BCa method, which is very accurate, but it is also computationally intensive. This method adjusts for bias and non-constant variance, and yields the percentile interval in the event that the sample is unbiased with constant variance.

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