met-8.0 Enhancements

MET 7.0 Bug Fixes

https://dtcenter.org/met/users/support/known_issues/METv7.0/index.php

09/24/2018: Fix equality checking for complex thresholds.
09/07/2018: Fix Grid-Stat slow runtimes for neighborhood methods.
07/24/2018: Fix Grid-Stat memory allocation issue causing slow runtimes.
06/13/2018: Fix user-reported compilation errors.
05/18/2018: Fix logic when applying multiple censoring thresholds.
05/18/2018: Fix logic when sizing the ascii output for Point-Stat and Grid-Stat.
05/03/2018: Fix tc_pairs interp12 logic for ATCF ID's ending with "32".
04/26/2018: Fix processing of GRIB2 MXUPHL.
04/03/2018: Fix ORANK header columns.
03/27/2018: Fix STAT-Analysis conversion from VL1L2 to VCNT.
03/27/2018: Fix PB2NC for large input files.
03/20/2018: Fix uninitialized value bug in grid_stat.

MET 8.0 Major Enhancements

- https://dtcenter.org/met/users/support/release_notes/METv8.0_release_notes.php
- Python Embedding
- Observation Error Logic in Ensemble-Stat
- Shapefiles for masking regions (gis tools)
- Point observation format overhaul
  - Time summaries
- Rotated Lat/Lon projection
- MODE object filters

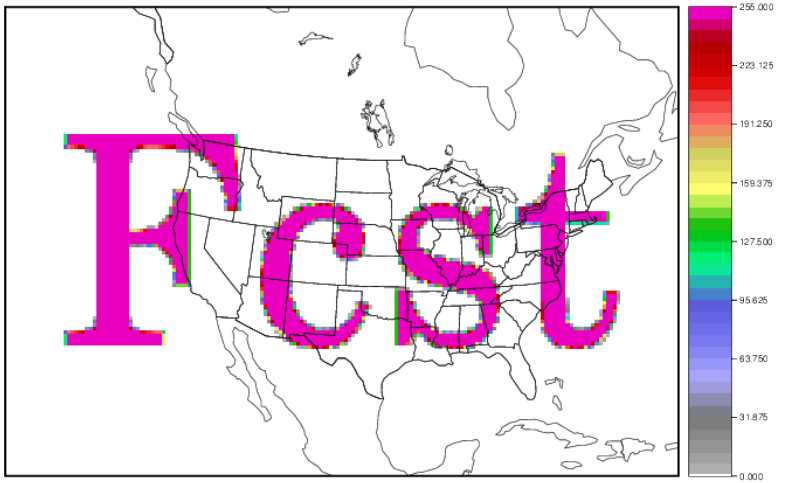
Python Embedding - Script

- User writes a script to read gridded data into a 2-dimensional array of data.
- May use NumPy (tested and described here) or Xarray (untested).
- The 2D NumPy array must be named **met_data**.
- The script defines a dictionary named **attr** which defines:
  - **valid** and initialization (**init**) times as strings in YYYYMMDD[_HH[MMSS]] format.
  - **lead** and accumulation (**accum**) times as strings in HH[MMSS] format.
  - **name**, **long_name**, level, and **units** as strings.
  - **grid** dictionary defining the projection and grid information in the same way as the gridded NetCDF files produced by MET.
- Python script may use a combination of environment variables and/or command line arguments.
- On the command line for the MET tools, replace the path to a gridded data file with the string **PYTHON_NUMPY** or **PYTHON_XARRAY**.
- In the **field** string, set **name** to the Python command to be executed.

Python Embedding - Example

- Sample script and data included in release.
- read_ascii_numpy.py reads data from ASCII files.
- Make sure this Python command runs well.
  - **cd met-8.0**
  - **python scripts/python/read_ascii_numpy.py data/python/fcst.txt FCST**
- Run **plot_data_plane** to read data from a Python script and plot it.
- Usage: plot_data_plane
  - input_filename = gridded data file
  - output_filename = PostScript plot file
  - field_string = description of data to plot

plot_data_plane PYTHON_NUMPY python.ps 'name="scripts/python/read_ascii_numpy.py data/python/fcst.txt FCST";'



Python Embedding - MET Config

- May replace gridded data file with **PYTHON_NUMPY** or **PYTHON_XARRAY** on the command line for all of the MET tools.
- Grid-to-Grid tools (like Grid-Stat and MODE) may set one or both to Python.
- Grid-Stat Example:
  - **grid_stat PYTHON_NUMPY PYTHON_NUMPY GridStatConfig_python**

```
fcst = {
   field = [ { name = "scripts/python/read_ascii_numpy.py
                       data/python/fcst.txt FCST"; } ];
}
obs = {
   field = [ { name = "scripts/python/read_ascii_numpy.py
                       data/python/obs.txt OBS"; } ];
}
```

Python Embedding - Future

- MET calling Python:
  - Further testing of the **PYTHON_XARRAY** option.
  - Enhance logic to read *multiple vertical levels* from Python (e.g. run Point-Stat or Ensemble-Stat to verify multiple pressure levels)
  - Enhance logic to read a *time series of gridded data* from Python (e.g. for Series-Analysis and MTD tools)
  - Enhance logic to read *point observations* from Python (e.g. for Point-Stat and Ensemble-Stat)
- Python calling MET:
  - Define entry points for Python scripts to *call MET tools* directly.

Observation Error - Logic

- Account for observation error when verifying ensemble forecasts.
- Incorporate the method used by Meteo France into the Ensemble-Stat tool.
- May be applied to point observations or gridded analyses.
- Each observation value is compared to N ensemble member values.
- For each observation value, locate the observation error information (defined in config file or via a table lookup).
- Observation error information defines bias correction information and the assumed error distribution.
- ***For each observation value…***
  - Apply any bias correction logic.
- ***For each ensemble member value…***
  - Randomly sample from the error distribution centered on 0.
  - Add random perturbation to current ensemble member value.
  - Apply min/max criteria to keep perturbed value within valid physical limits.
- Perturbations are applied to the ensemble forecast values to make them comparable with the observations which already include errors.

Observation Error - MET Config

- Ensemble-Stat config file turns observation error perturbations ON/OFF.
- Applies the *same error distribution* to all observations for this task.

```
// Observation error options
// Set dist_type to NONE to use the observation error table instead
// May be set separately in each "obs.field" entry
obs_error = {
   flag             = FALSE;   // TRUE or FALSE
   dist_type        = NONE;    // Distribution type (e.g. NORMAL)
   dist_parm        = [];      // Distribution parameters (e.g. 2.0)
   inst_bias_scale  = 1.0;     // Instrument bias scale adjustment
   inst_bias_offset = 0.0;     // Instrument bias offset adjustment
   min              = NA;      // Valid range of data
   max              = NA;
}
```

- Bias Correction: obs_bc = obs * inst_bias_scale + inst_bias_offset;

Observation Error - Table Lookup

- Observation error table lookup provides much finer control of the error distributions.
- In Ensemble-Stat config file, set **"flag = TRUE;"** and **"dist_type = NONE;"**
- If **MET_OBS_ERROR_TABLE** is set, read that table.
- If unset, read defaults from **MET_BASE/table_files/obs_error_table.txt**.
- Columns of observation error table file:
  - Filter criteria to find a match:
    - **OBS_VAR**: Comma-separated list of regular expressions for variable strings to match
    - **MESSAGE_TYPE**: Comma-separated list of message types to match
    - **PB_REPORT_TYPE, IN_REPORT_TYPE, INSTRUMENT_TYPE**: Additional filters
    - **STATION_ID**: Comma-separated list of Station ID's to match
    - **HGT_RANGE, PRS_RANGE, VAL_RANGE**: Numeric filter range. Set to **ALL** or min,max
  - Observation bias correction settings:
    - **INST_BIAS_SCALE**: Observation bias scale value (multiply)
    - **INST_BIAS_OFFSET**: Observation bias offset value (add)
  - Random perturbation settings:
    - **DIST_TYPE**: Distribution type (see User's Guide)
    - **DIST_PARM**: Parameters of distribution
    - **MIN, MAX**: Physical bounds for this variable

Observation Error - Example

- Default perturbations in obs_error_table.txt provided by Jeff Beck (NOAA/GSD) based on literature review.
- Defaults included for surface (**ADPSFC**) and upper-air (**ADPUPA**) message types for **TMP**, **DPT**, **WIND**, **UGRD**, and **VGRD**, as well as **HGT** and **APCP**.

```
APCP_[0-9]* ALL ALL ALL ALL ALL ALL ALL 0,0.01  NA NA NONE   NA   0 NA
APCP_[0-9]* ALL ALL ALL ALL ALL ALL ALL 0.01,10 NA NA NORMAL 0.1  0 NA
APCP_[0-9]* ALL ALL ALL ALL ALL ALL ALL 10,50   NA NA NORMAL 1.0  0 NA
APCP_[0-9]* ALL ALL ALL ALL ALL ALL ALL 50,200  NA NA NORMAL 5.0  0 NA
APCP_[0-9]* ALL ALL ALL ALL ALL ALL ALL 200,500 NA NA NORMAL 15.0 0 NA
```

- **APCP_[0-9]*** uses regular expressions to match multiple accumulations.
- Lines match **ALL** message, pb report, input report, and instruments types.
- Lines match **ALL** station ID's and height and pressure ranges.
- Define different perturbations based on the value ranges, increasing from **NONE** for 0 - 0.1 mm up to **NORMAL(15)** for 200 - 500 mm.
- Min/Max values (**0 NA**) prevent negative perturbations.
- Increase Ensemble-Stat verbosity level (**-v 4**) to see perturbation details.

Observation Error - Output

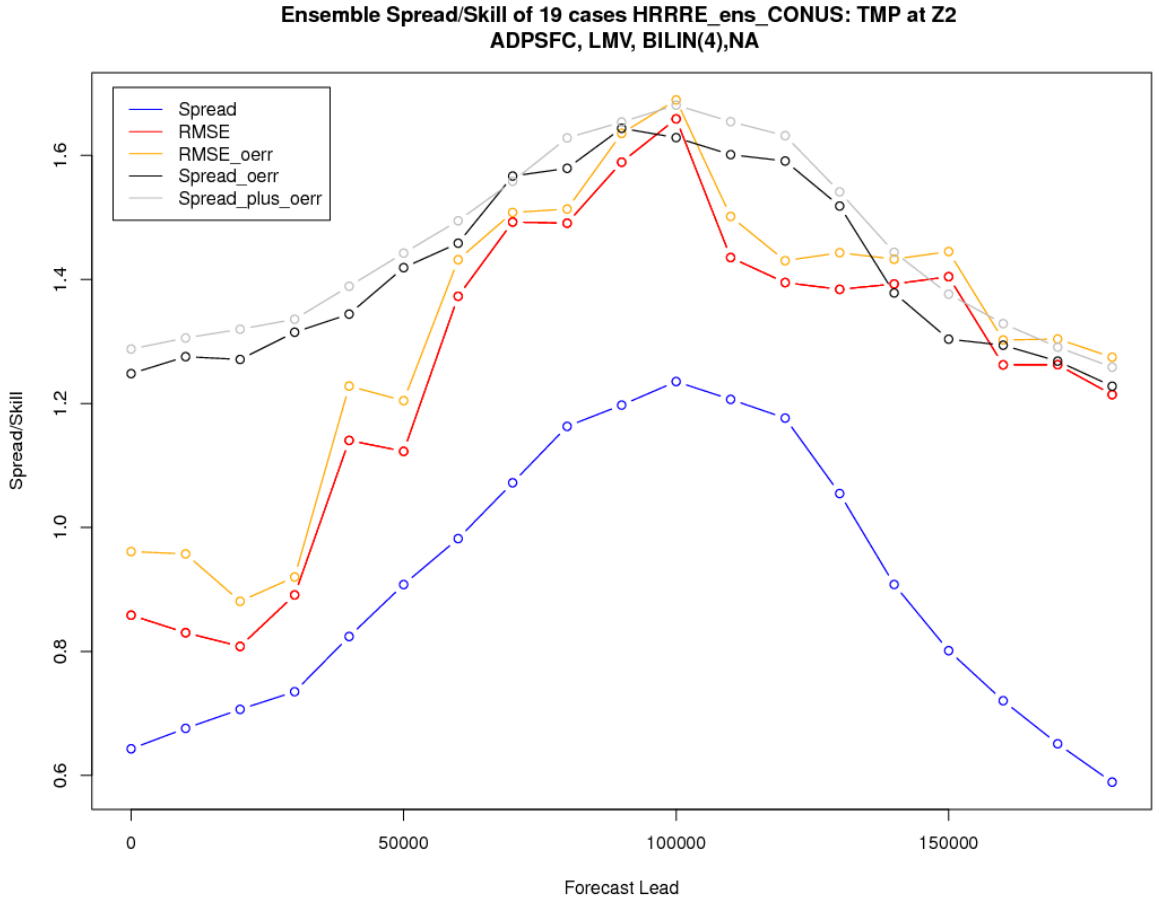
- Add new Ensemble Continuous Statistics (**ECNT**) line type.
- Move columns for CRPS, IGN, CRPSS, and SPREAD from **RHIST** to new **ECNT** line type.

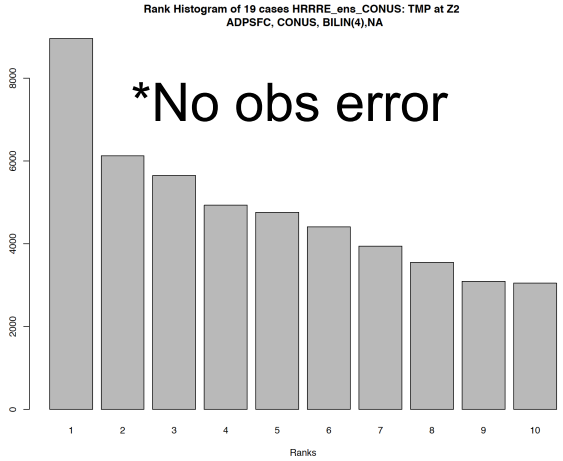
Table 9.2: Format information for ECNT (Ensemble Continuous Statistics) output line type.

ECNT OUTPUT FORMAT		
Column Number	**ECNT Column Name**	**Description**
22	ECNT	Ensemble Continuous Statistics line type
23	TOTAL	Count of observations
24	N_ENS	Number of ensemble values
25	CRPS	The Continuous Ranked Probability Score
26	CRPSS	The ContinuousRanked Probability Skill Score
27	IGN	The Ignorance Score
28	ME	The Mean Error of the ensemble mean (unperturbed or supplied)
29	RMSE	The Root Mean Square Error of the ensemble mean (unperturbed or supplied)
30	SPREAD	The mean of the spread (standard deviation) of the unperturbed ensemble member values at each observation location
31	ME_OERR	The Mean Error of the PERTURBED ensemble mean (e.g. with Observation Error)
32	RMSE_OERR	The Root Mean Square Error of the PERTURBED ensemble mean (e.g. with Observation Error)
33	SPREAD_OERR	The mean of the spread (standard deviation) of the PERTURBED ensemble member values (e.g. with Observation Error) at each observation location
34	SPREAD_PLUS_OERR	The square root of the sum of unperturbed ensemble variance and the observation error variance

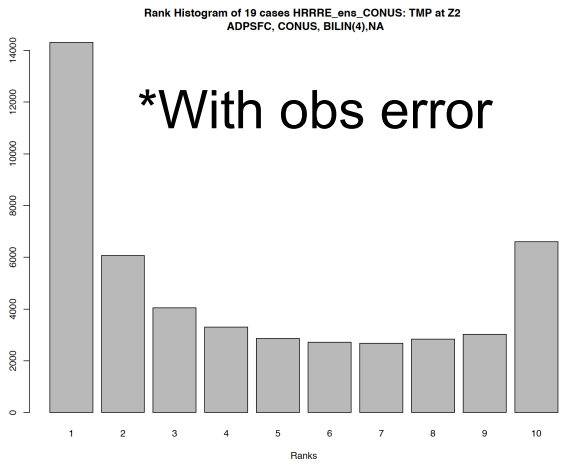
Observation Error - Plots

- When **obs_error** is enabled, perturbed data is used for all output line types:
  - ECNT, RHIST, PHIST, ORANK, SSVAR, RELP





*No obs error



*With obs error

* Images courtesy of Jeff Beck (NOAA/GSD and DTC)

Shapefiles - Masking Regions

- Enhance **gen_vx_mask** by adding support for "**-type shape**" and "**-shapeno n**" command line options.
- Read closed polygon shapefiles (not points or lines) to define regions of interest.
- Many source for ESRI shapefiles, including:
  - Natural Earth are freely available at low, medium, and high resolution:
    - https://www.naturalearthdata.com/downloads
  - NWS River Forecast Centers and Basins:
    - https://www.nohrsc.noaa.gov/gisdatasets
- GIS datasets consist of 3 mandatory files:
  - **.shp** - shape format; the feature geometry itself
  - **.shx** - shape index format; a positional index of the feature geometry to allow seeking forwards and backwards quickly
  - **.dbf** - attribute format; columnar attributes for each shape

Shapefiles - gis utilities

- The new **gis_dump_dbf** (metadata), **gis_dump_shp** (data), and **gis_dump_shx** (indexes) utilities dump information about these shapefiles.
- Example of contents for River Forecast Centers.

gis_dump_dbf rf12ja05.dbf

Record 0 ...
| SITE_ID = "ACR"
| STATE = "AK"
| RFC_NAME = "Alaska"
| RFC_CITY = "Anchorage"
| BASIN_ID = "AKRFC"

Record 1 ...
| SITE_ID = "KRF"
| STATE = "MO"
| RFC_NAME = "Missouri Basin"
| RFC_CITY = "Kansas City/Pleasant Hill"
| BASIN_ID = "MBRFC"

Record 2 ...

gis_dump_shp rf12ja05.shp

| points = [
| | (-131.534, 55.29)
| | (-131.534, 55.29)
| | (-131.534, 55.29)
| | (-131.539, 55.2892)

gis_dump_shx rf12ja05.shx

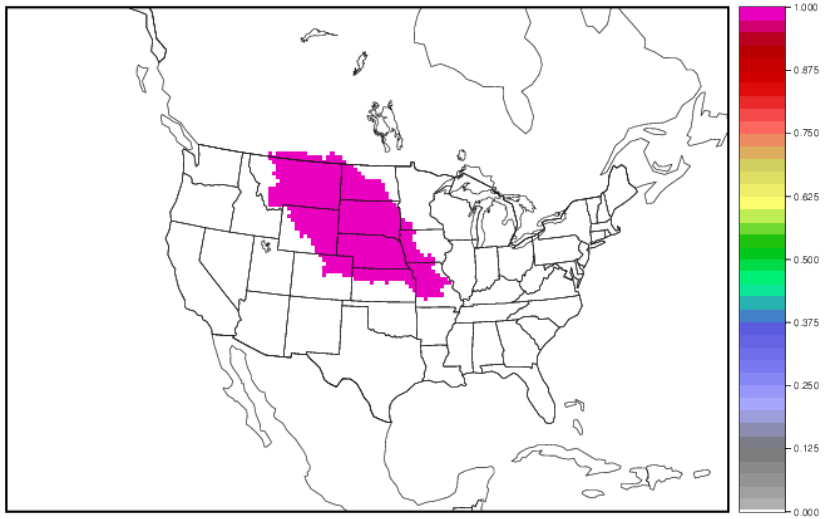
Record 0 ...
| offset = 100
| content_length = 4037176

Record 1 ...
| offset = 4037284
| content_length = 339984

Record 2 ...

Shapefiles - gen_vx_mask

- Run gen_vx_mask to define the MBRFC River Forecast Center.
- Appears in the 2nd record, so set 0-based "**-shapeno 1**".



G212_MBRFC.nc

**/usr/local/met-8.0/bin/gen_vx_mask **

**G212.grb rf12ja05.shp G212_MBRFC.nc **

-type shape -shapeno 1 -name MBRFC -v 3

DEBUG 1: Input File:	G212.grb
DEBUG 1: Mask File:	rf12ja05.shp
DEBUG 2: Parsed Data Grid:	Lambert Conformal (185 x 129)
DEBUG 2: Parsed Shape Mask:	rf12ja05.shp containing 21246 points
DEBUG 3: Shape Masking:	914 of 23865 points inside
DEBUG 1: Output File:	G212_MBRFC.nc

GOES-16 AOD

- Enhance **regrid_data_plane** to read GOES-16 AOD data and regrid to a user-specified regular grid.

**regrid_data_plane **

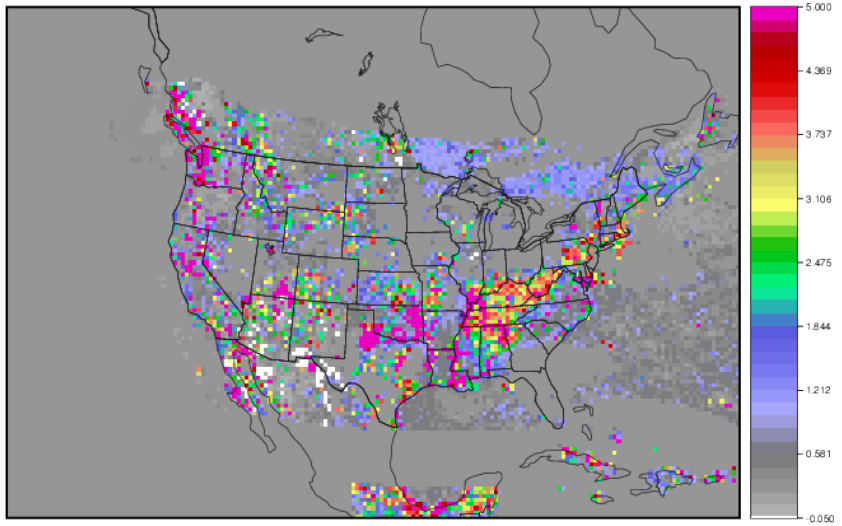
**OR_ABI-L2-AODC-M3_G16_s20181341702215_e20181341704588_c20181341711418.nc **

**G212 GOES-16_AOD_TO_G212.nc **

**-field 'name="AOD"; level="(*,*)";' **

**-coord goes_16/g16_conus_latlon_2km_20180620.dat **

-qc 1,2,3 -method MAX -v 2



Point Obs - Time Summaries

- The ASCII2NC and PB2NC tools support the **time_summary** config file option.
  - In met-7.0, time summaries are computed separately for each input file.
  - In met-8.0, time summaries are computed across all input files.
  - In met-8.0, add **time_summary.raw_data** = TRUE/FALSE.
    - TRUE means write raw input values *AND* time summary values.
    - FALSE means only write the time summary values.
  - In met-8.0, allow **time_summary.width** to be set to an integer (for centered time window) or a dictionary (for un-centered time window)
    - Hourly summary for +/- 30 minutes
      - **time_summary.width = 3600;**
    - Hourly summary for the previous hour:
      - **time_summary.width = { beg = -3600; end = 0; }**

Additional Changes

- Refine NetCDF point observation format to store strings more efficiently.
  - Store arrays of unique strings.
  - For each observation, store array offsets for the strings.
- Read rotated lat/lon projections from GRIB2, Python, and MET-NetCDF files.
- In **MODE**, add **filter_attr_name** and **filter_attr_thresh** config file options to filter objects by many attributes.
  - Remove deprecated **area_thresh** and **inten_perc_thresh** options.
  - Add **aspect_diff** and **curvature_ratio** as fuzzy logic inputs and write them to the output files.
- In **Point-Stat** and **Ensemble-Stat**, add new **mask.llpnt** config file option to apply thresholds to the lat/lon of point observations.
  - Enables Point-Stat to exactly match the number of matched pairs of VSDB Global Grid-to-Obs tool.
  - Effectively limited to defining rectangular regions.
- In Point-Stat, enhance the **HiRA** neighborhood verification methodology to write the **ECNT** line type.
- Switch to storing masking regions as booleans instead of double-precision.

METviewer Enhancements

METviewer Release History

- METviewer release cycle is quick and responsive.
- Distribute bugfixes as a hotfix for the current release or in a new release.

http://www.dtcenter.org/met/metviewer/doc/versions.html

- **mv2.4** – 03/13/2018
  - Update for met-7.0 changes, add vector statistics for winds, read MTD output, scorecard improvements, and code refactoring
- **mv2.5** – 05/08/2018
  - Add contour plot type and scorecard improvements
- **mv2.6** – 08/03/2018
  - Add CI's to reliability diagrams, revision series, and organize databases into groups
- **mv2.7** – 08/31/2018
  - Vector wind improvements and scorecard updates
- **mv2.8** – 09/31/2018
  - Update for met-8.0 changes (move columns from RHIST to ECNT line type and new columns of MODE output)

mv2.4 - Vector Stats

- **Appendix G** of the MET Users Guide lists the equations for the vector statistics in the VCNT line type.

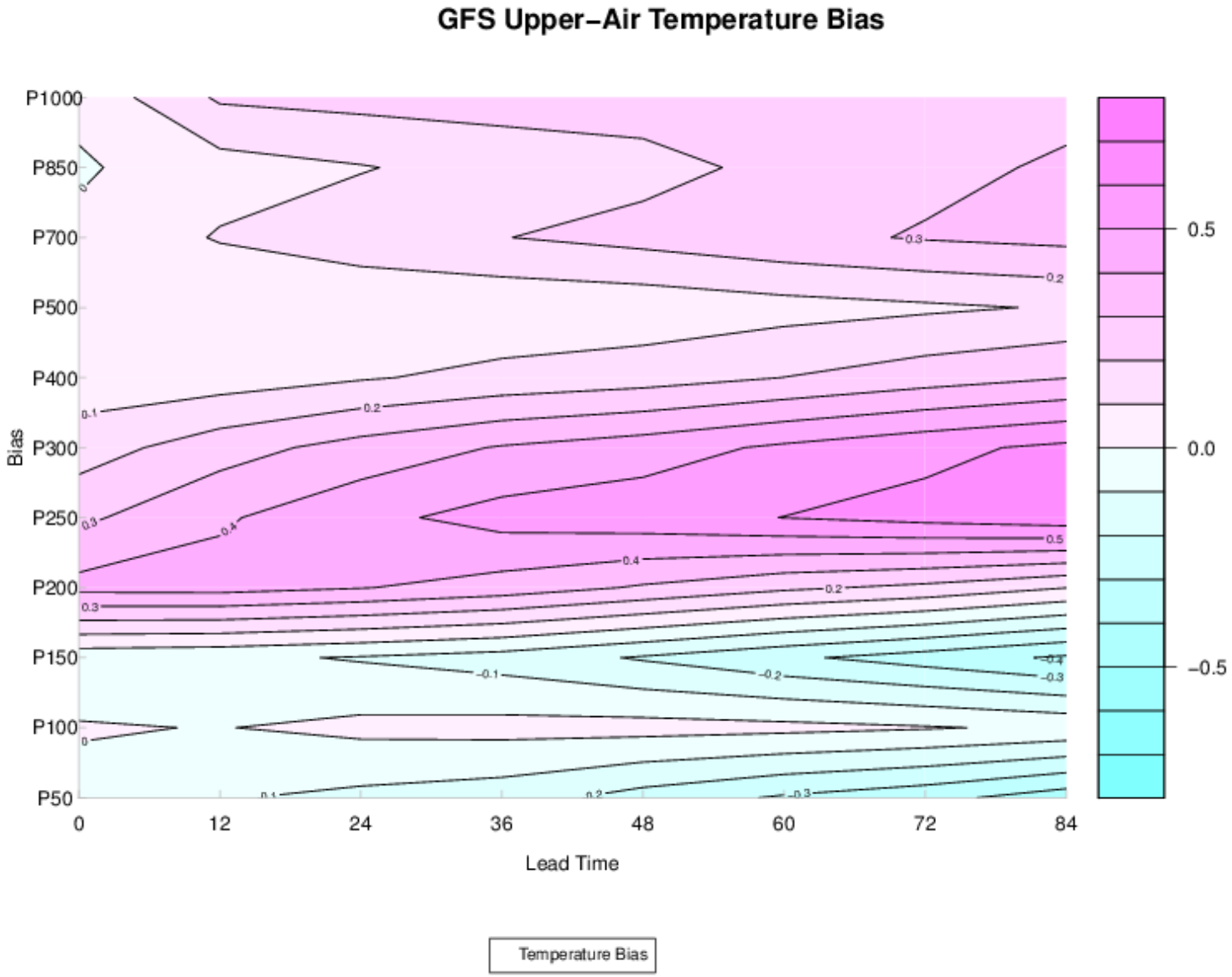
VCNT OUTPUT FORMAT		
Column Numbers	**VCNT Column Name**	**Desc**
23	TOTAL	Total numbe
24–26	FBAR	Mean value of f
27–29	OBAR	Mean value of o
30–32	FS_RMS	Root mean square
33–35	OS_RMS	Root mean square
36–38	MSVE	Mean squared length of the forecast and
39–41	RMSVE	Square root of MSVE
42–44	FSTDEV	Standard deviation of the forecast wind speed
45–47	OSTDEV	Standard deviation of the observed wind field
48–50	FDIR	Direction of the average forecast wind vector
51–53	ODIR	Direction of the average observed wind vector
54–56	FBAR_SPEED	Length (speed) of the average forecast wind vector
57–59	OBAR_SPEED	Length (speed) of the average observed wind vector
60–62	VDIFF_SPEED	Length (speed) of the vector difference between the average forecast and average observed wind vectors
63–65	VDIFF_DIR	Direction of the vector difference between the average forecast and average wind vectors
66–68	SPEED_ERR	Difference between the length of the average forecast wind vector and the average observed wind vector (in the sense F - O)
69–71	SPEED_ABSERR	Absolute value of SPEED_ERR
72–74	DIR_ERR	Signed angle between the directions of the average forecast and observed wing vectors. Positive if the forecast wind vector is counterclockwise from the observed wind vector
75–77	DIR_ABSERR	Absolute value of DIR_ABSERR

Appendix G

Vectors and Vector Statistics

In this appendix, we discuss some basic properties of vectors, concentrating on the two-dimensional case. To keep the discussion simple, we will assume we are using a Cartesian coordinate system.

mv2.5 - Contour Plot



mv2.3 - Economic Value Plot

- Computed from 2x2 CTC contingency tables or Nx2 PCT tables