NCAR

METplus - a Python Wrapped Verification Capability Unifying the US Verification and Validation Community

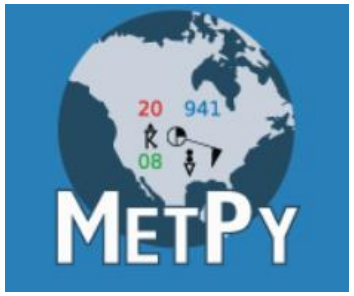
Presenter: Tara Jensen

Team: John Halley Gotway, Julie Prestopnik, Minna Win-Gildenmeister, Dan Adriaansen, Mallory Row, Perry Shafran, Jim Frimel, George McCabe, Howard Soh, Tatiana Burek, Randy Bullock ,Tina Kalb, Hank Fisher, and Jonathan Vigh

Oct 30-31, 2018
ECMWF Python Workshop for Earth System Science
Reading, UK

National Center for Atmospheric Research

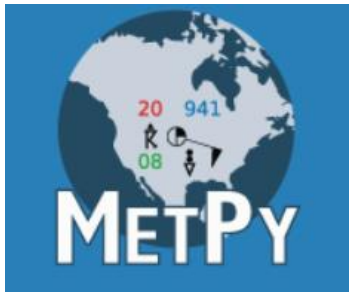
Namespace issues: We've heard about

Metview  and 

What is different about METplus

- Its more emphatic!!
- Name began as an acronym for Model Evaluation Tools (MET) and has expanded to METplus
- Initial focus was on computation of statistics and is growing to provide python workflow and visualization
- Has a component called METviewer database and display system

Namespace issues: We've heard about

Metview  and 

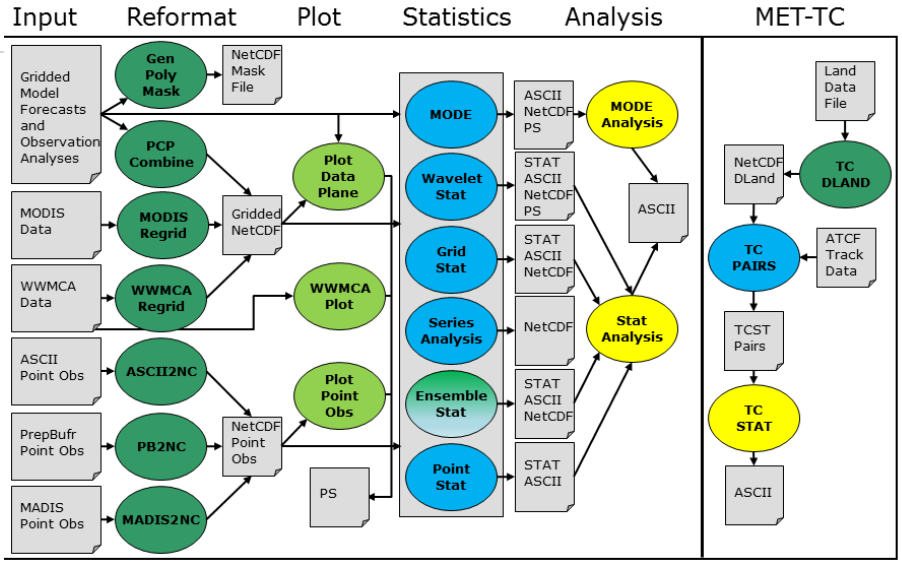
What is similar about METplus

- Similar to Metview – it's a suite of python scripts
- MET needs to be installed separately in environment
- Similar to Metview in that it computes verification scores and plotting both scores and meteorological fields
- Will rely on MetPy for plotting capability once it gets to 1.0

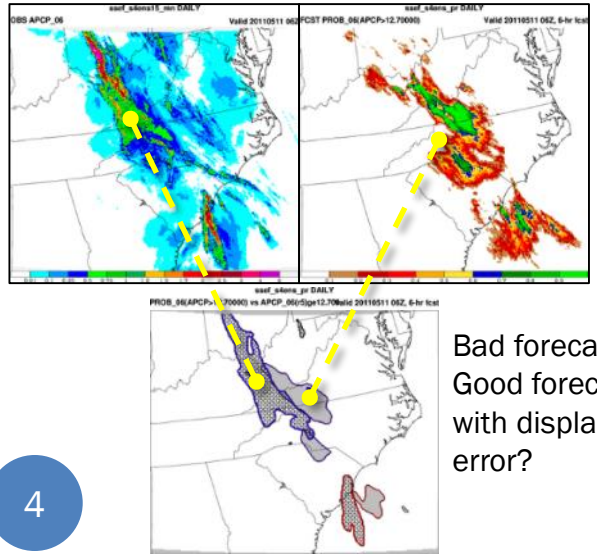
MET
Model Evaluation Tools

A Verification Toolkit Designed for Flexible Yet Systematic Evaluation (supported to the community via the DTC)

- Originally developed to replicated the EMC mesoscale verification system
- **Over 85 traditional statistics** using both point and gridded datasets
- 15 interpolation methods
- C++ with calls to Fortran Libraries
- Able to read in GRIB1, GRIB2 and CF-compliant NetCDF
- **Applied to many spatial and temporal scales**
- 3500+ users, both US & Int'l

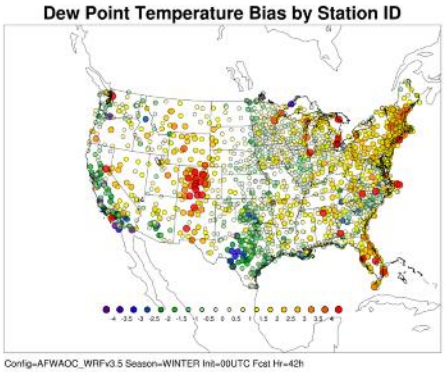


Object Based and Spatial Methods

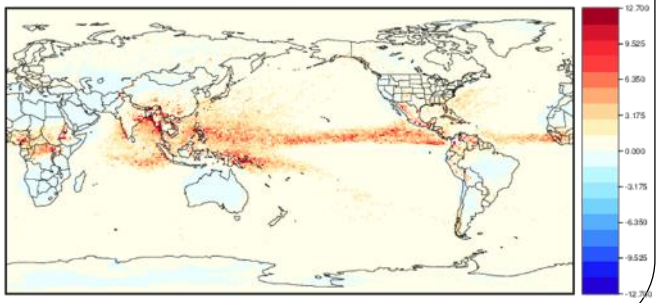


Bad forecast or Good forecast with displacement error?

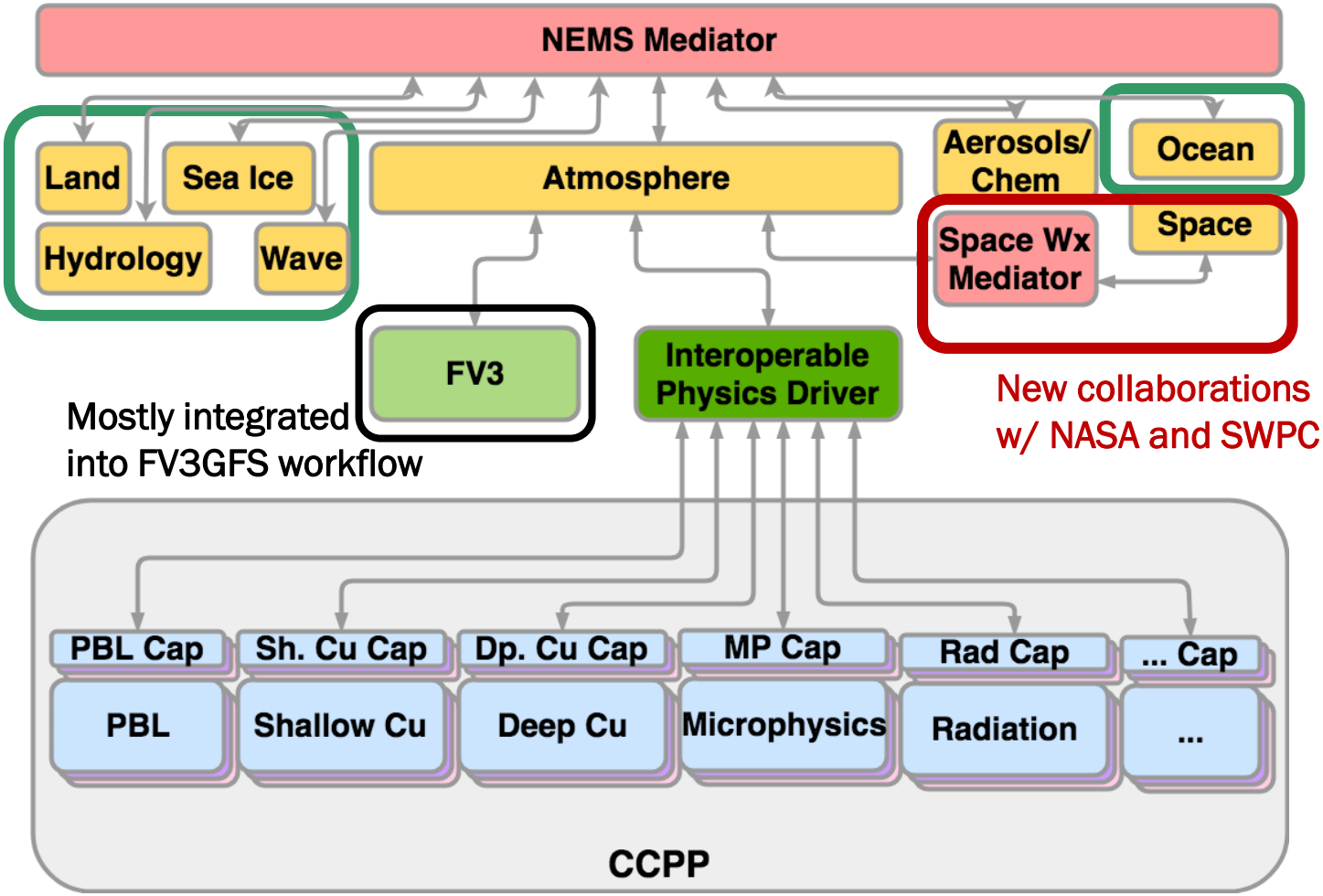
Geographical Representation of Errors



90th Percentile of difference between two models



The Goal: Have a Unified Capability to Evaluate All Aspects of the Coupled UFS*



Already have pre-existing packages which we will link in

Mostly integrated into FV3GFS workflow

New collaborations w/ NASA and SWPC

Image courtesy of Global Model Test Bed within the Developmental Testbed Center

*Unified Forecast System (UFS) is the US NWS next generation Earth system model

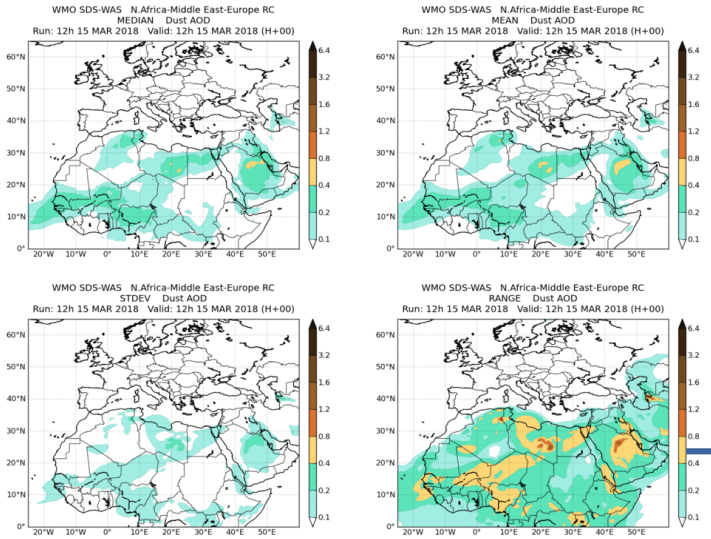
Working Towards Easy Usability: METplus Use Case Example

Observed 1-min AOD Data

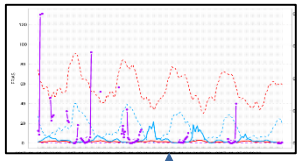
ASCII 2NC

Observed AOD 6-hr mean, max, stdev, range

Forecasted Aerosol Optical Depth (AOD): 6-hr mean, max stdev, range



- Wavelet Stat
- MODE-TD
- MODE
- Grid Stat
- Series Analysis
- Ensemble Stat
- Point Stat



Plotting Script

STAT ASCII

Stat Analysis

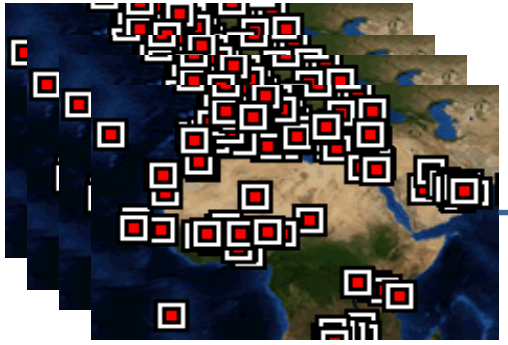
STAT ASCII

Working Towards Easy Usability: METplus Use Case Example

Observed 1-min AOD Data

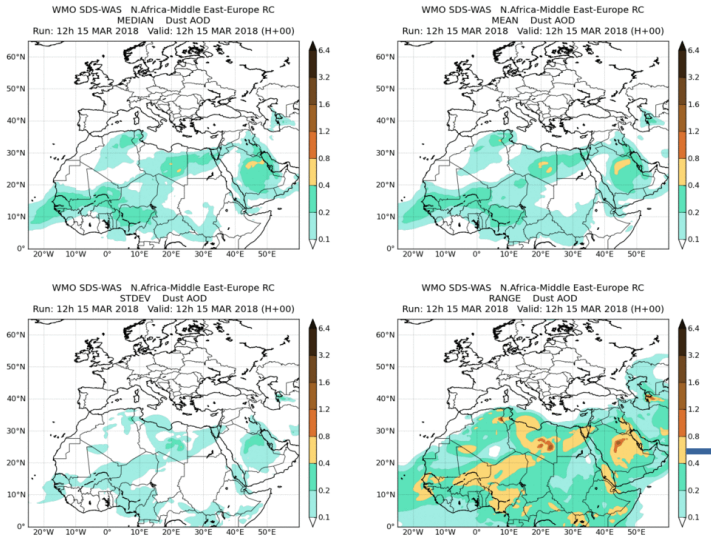
ASCII 2NC

Observed AOD 6-hr mean, max, stdev, range

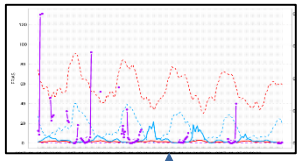


Use-case includes

- METplus .conf file
- MET config files
- Python scripts to:
  - Call Ascii2NC
  - Call Point-Stat
  - Call Stat-Analysis
  - Make statistics plot
  - Make plot of fields



Wavelet Stat
MODE-TD
MODE
Grid Stat
Series Analysis
Ensemble Stat
Point Stat



Plotting Script

STAT ASCII

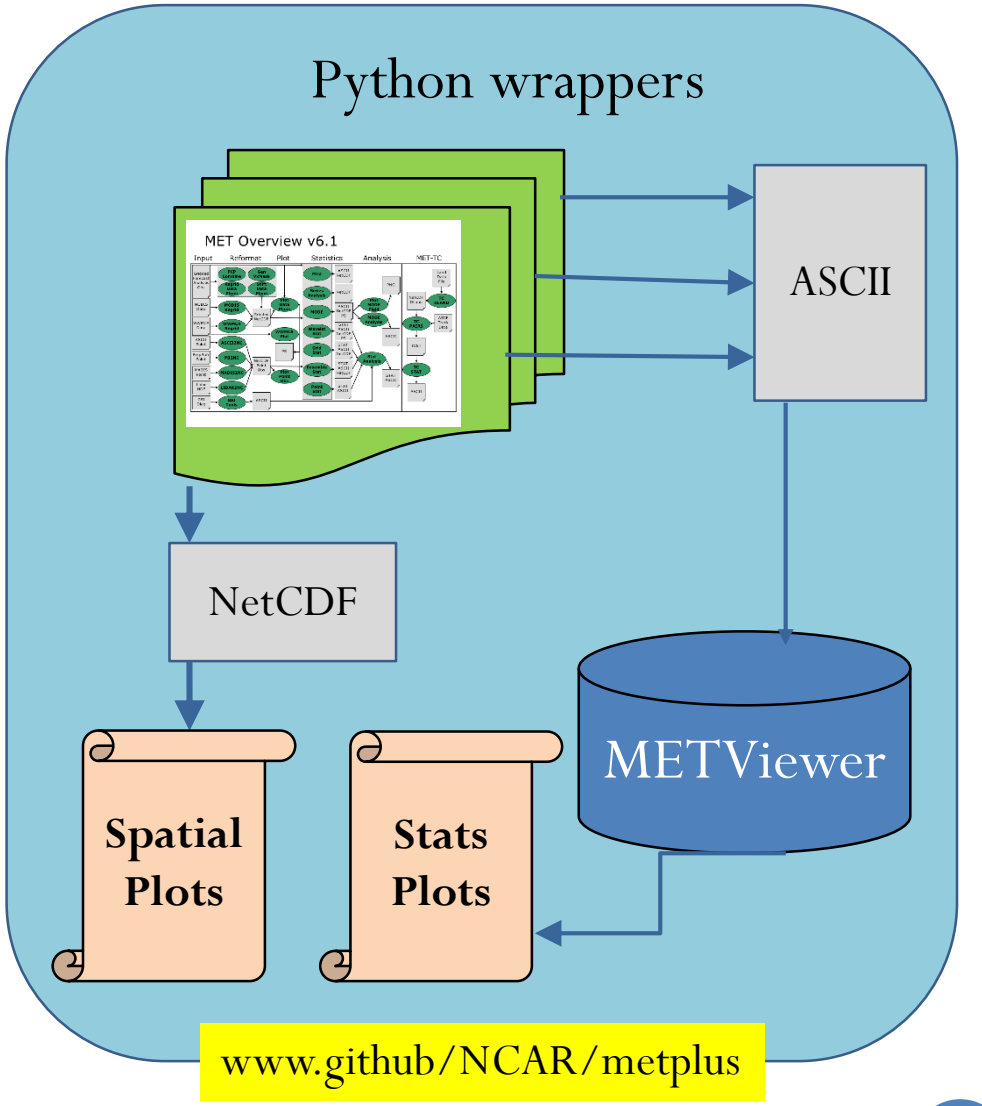
Series Analysis

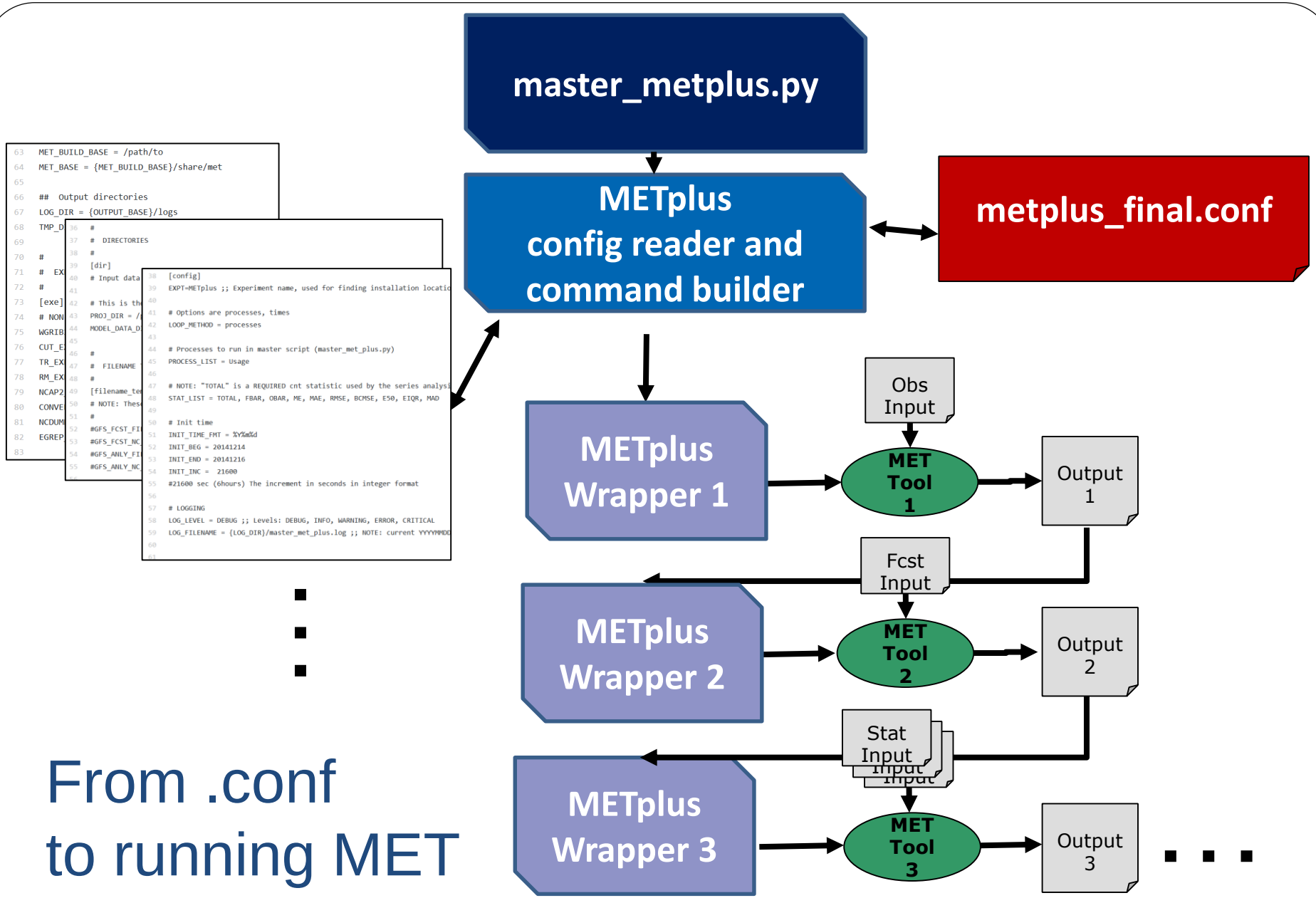
STAT ASCII

General Concept of METplus

Python wrappers around:

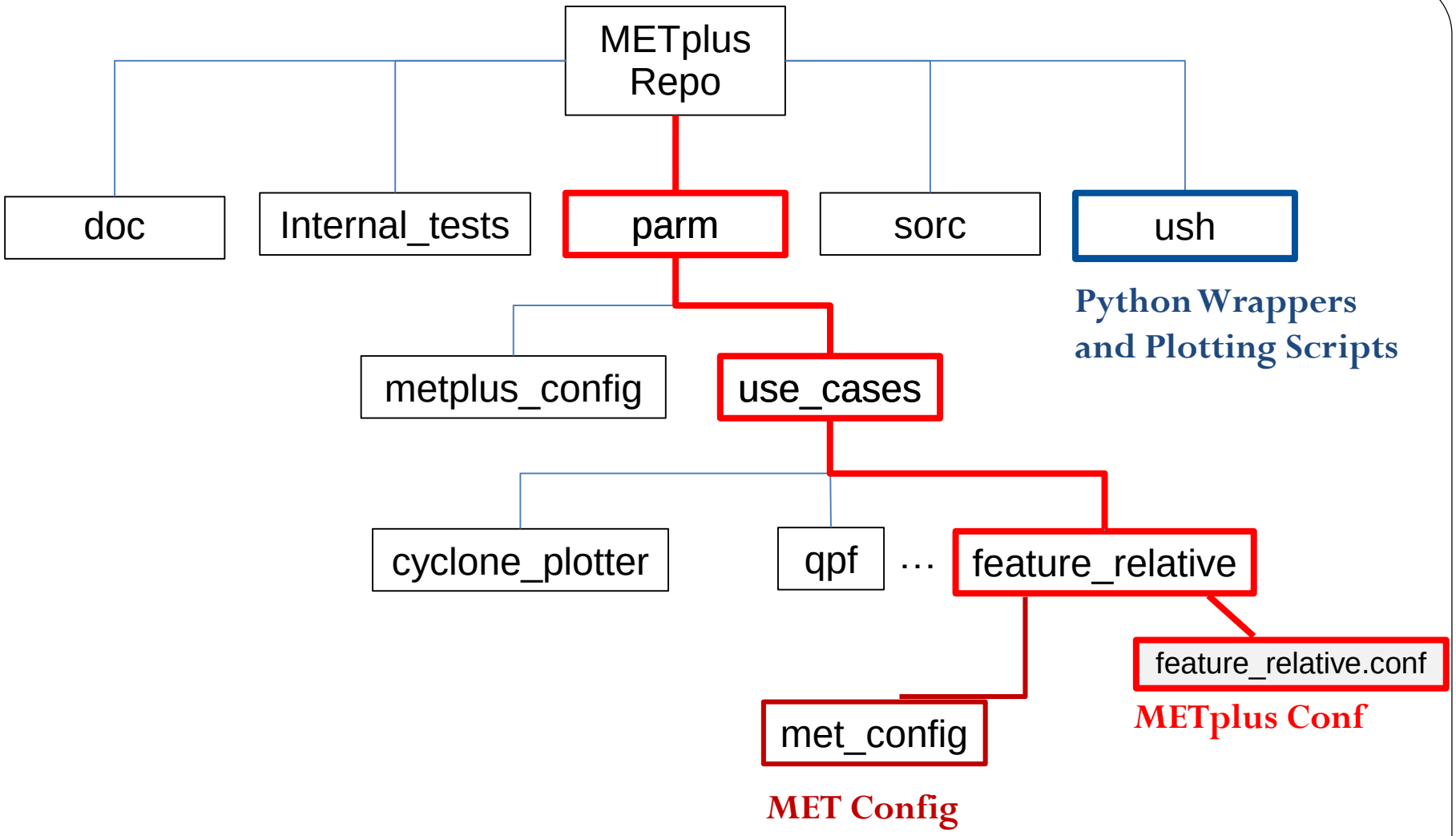
- MET (core)
- METviewer database and display (core)
- Plotting
  - METviewer User Interface
  - METviewer Batch Engine
  - **Python plotting scripts**
- **Communication between MET & python algorithms**





From .conf
to running MET

Flexible Configuration


METplus Repo
doc
Internal_tests
parm
sorc
ush
Python Wrappers and Plotting Scripts
metplus_config
use_cases
cyclone_plotter
qpf
…
feature_relative
feature_relative.conf
METplus Conf
met_config
MET Config

What does wrapped by Python mean?

METplus/parm/use_cases/feature_relative

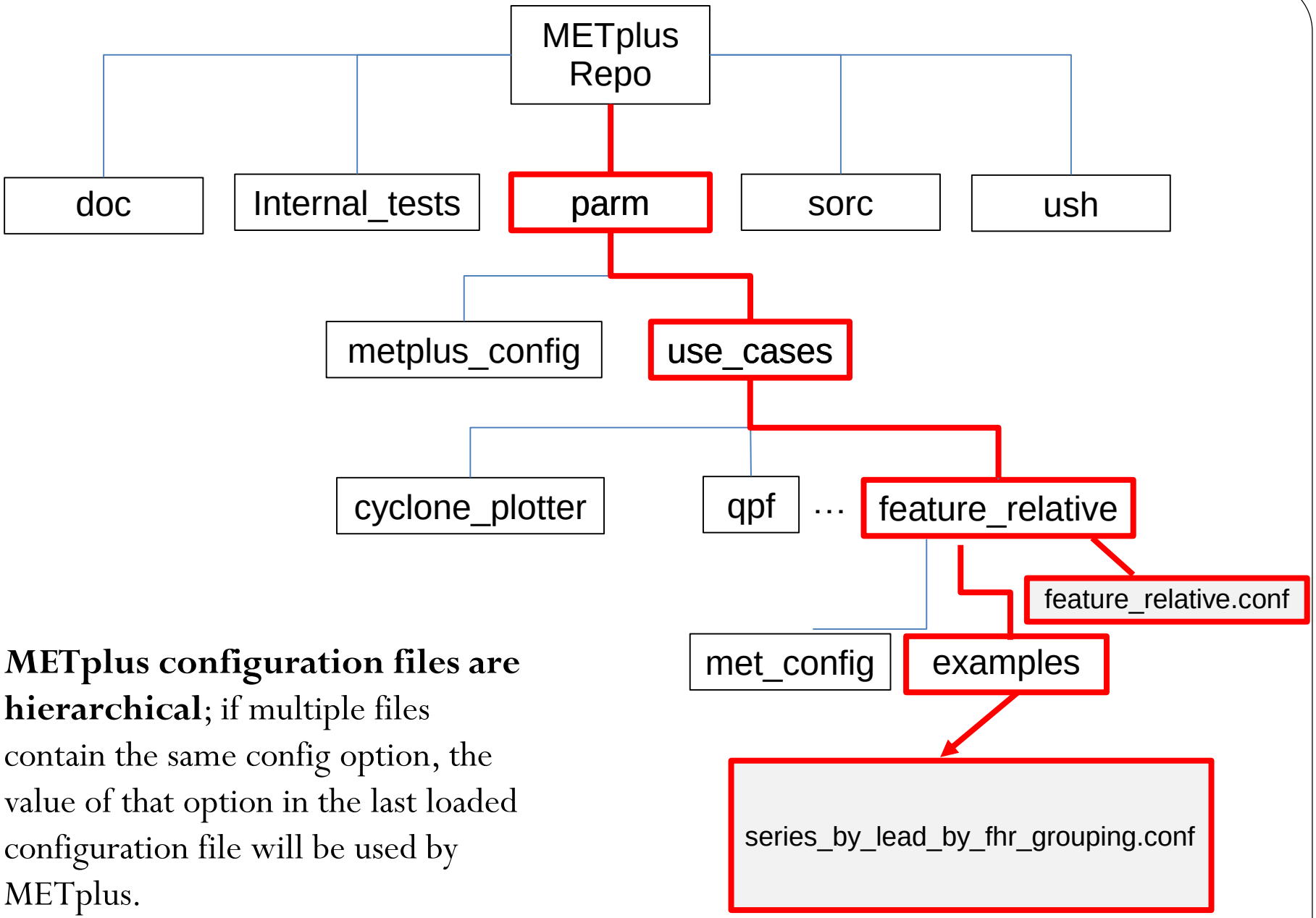
METplus Conf

feature_relative.conf

```
#
#     LISTS AND SETTINGS
#

#     Processes to run in master script (master_met_plus.py)

PROCESS_LIST = ["run_tc_pairs.py", "extract_tiles.py", "series_by_lead.py"]

#
#     NOTE: "TOTAL" is a REQUIRED cnt statistic used by the series analysis scripts
#

STAT_LIST = ["TOTAL", "FBAR", "OBAR", "ME", "MAE", "RMSE", "BCMSE", "E50", "EIQR", "MAD"]

#     Dates must be in YYYYMMDD format
#     INIT_HOUR_INC is the increment in integer format
#     INIT_HOUR_END should be a string in HH or HHH format

INIT_DATE_BEG = "20141201"
INIT_DATE_END = "20150331"
INIT_HOUR_INC =  6
INIT_HOUR_END = "18"

#     Used by extract_tiles.py to define the records of interest from the grib2 file

VAR_LIST = ["HGT/P500", "PRMSL/Z0", "TMP/Z2", "PWAT/L0", "HGT/P250", "TMP/P850", "TMP/P500", "UGRD/P250", "VGRD/P250" ]
EXTRACT_TILES_VAR_LIST = []

#     Used for performing series analysis based on lead time
```

MET Config

```
fcst = {

   field = [
      {
         name  = "${NAME}";
         level = [ "${LEVEL}" ];
      }
   ];
```



METplus configuration files are hierarchical; if multiple files contain the same config option, the value of that option in the last loaded configuration file will be used by METplus.

Configuration Files: Documentation

```
[config]
INIT_BEG=2018100100

[dir]
CONFIG_DIR={PARM_BASE}/use_cases/qpf/met_config

[filename_templates]
OBS_PCP_COMBINE_INPUT_TEMPLATE = {valid?fmt=%Y%m%d}/ST4.{valid?fmt=%Y%m%d%H}.{level?fmt=%HH}h
```

Q: How do I know what "family" ([dir], [config], etc…) a config option belongs to?

A: The METplus User Guide contains a "Config Glossary" that contains every METplus configuration option with various information including which family the config option belongs to.

Adding Flexibility by Embedding Python

Python Embedding - API

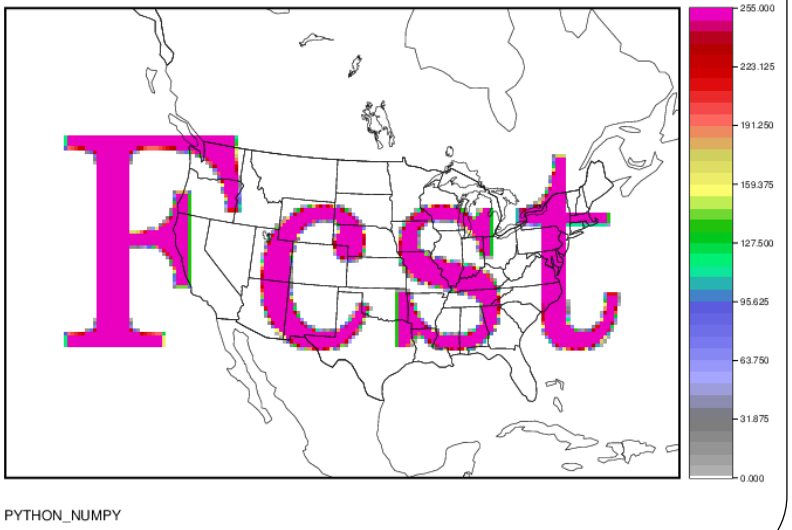
- Linking to the appropriate libraries similar to NetCDF, Grib, etc .
- Using Python's C API that is exported
- Include the header files in MET code and link to the libraries at compile time.
- C API online documentation (https://docs.python.org/2/c-api/index.html)
- Details of extending and/or embedding Python using C or C++ (https://docs.python.org/2/extending/index.html#extending-index)

- **Big Challenge:** Setting up the python runtime environment

Python Embedding - Example

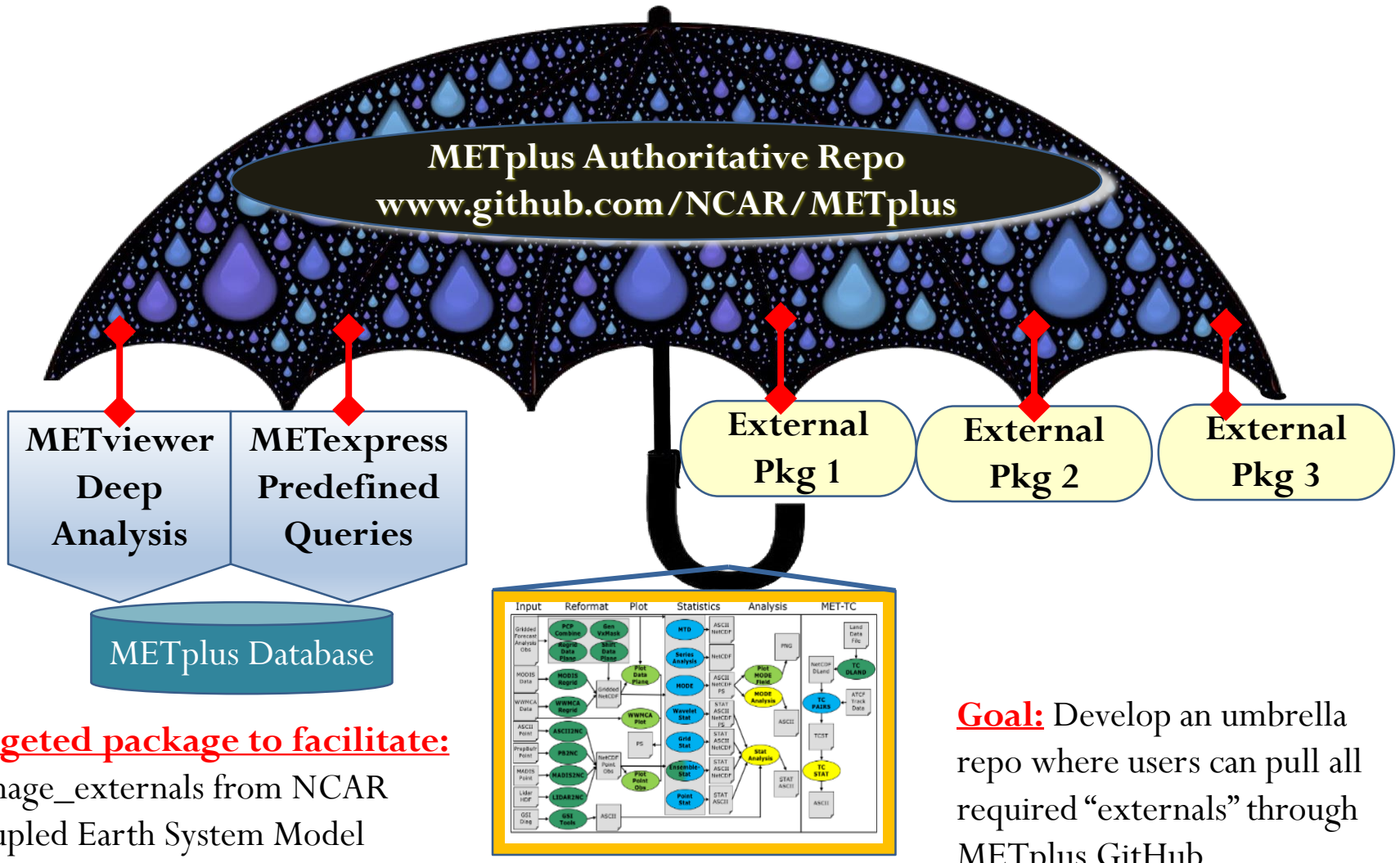
- Command Line
  - **python scripts/python/read_ascii_numpy.py data/python/fcst.txt FCST**
- MET config file setting
  - **plot_data_plane PYTHON_NUMPY python.ps 'name="scripts/python/read_ascii_numpy.py data/python/fcst.txt FCST";'**

Result of running the above sample script and data included in METv8.0 release



Future Work

Developing a Strategy for METplus Authoritative Repository



Targeted package to facilitate:
manage_externals from NCAR Coupled Earth System Model (CESM) framework

Goal: Develop an umbrella repo where users can pull all required "externals" through METplus GitHub.

Upcoming METplus Additions

MET component

- **C++ clean-up to pass cyber-security software scans (Fortify)** and improve memory handling and speed
- **Process Oriented Diagnostics for Subseasonal-to-Seasonal**
  - Moisture-Convection Coupling
  - MJO, NAO, and Teleconnection
  - TC Genesis
  - Extreme Weather related to Blocking
  - Cloud Property and Structure
  - Multi-variate fields and fluxes
- **CAM and Space Weather Evaluation**

Python Embedding - Future

- **MET calling Python:**
  - Further testing of the **PYTHON_XARRAY** option.
  - Enhance by adding **PYTHON_PANDA** logic to read *point observations* from Python (e.g. for Point-Stat and Ensemble-Stat)
  - 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)
- **Python calling MET:**
  - Define entry points for Python scripts to *call MET tools* directly.

Upcoming METplus Additions

- **Develop more use-cases** for Regional, Ensemble, Subseasonal to Seasonal forecast applications
- **Develop use-cases for Object-based Methods**
- **Add much more plotting capability** based on MetPy and other tools
- **Explore CDO, ODC, PANGEO, Dask** and other intriguing capabilities

Issues That We are Thinking About

- Running in the cloud and workflow in docker container – one big container or does python act as the glue layer around containers?
- Python layer queries the binaries to make a list of functions – how is this done?
- How best to manage transition from Python 2.7 to Python 3.x

- **NOTE to package developers:** Versioning matters! Especially to operational centers.
  - x.0 version is favorable to 0.x
  - Example 1.0 is better than 0.9 or 0.19

Contacts: Tara Jensen – jensen@ucar.edu and John Halley Gotway – johnhg@ucar.edu

METplus GitHub: github.com/NCAR/METplus

MET Users Page: www.dtcenter.org/met/users/

Container MET GitHub: github.com/NCAR/container-dtc-met

METviewer GitHub: github.com/NCAR/METviewer

Container METviewer Github: github.com/NCAR/container-dtc-metviewer

All help requests go through MET Helpdesk: ***met_help@ucar.edu***

METplus work is funded by the Developmental Testbed Center partners (NOAA, Air Force, NCAR and NSF), NGGPS program office, and USWRP R2O grants