Support for MET is provided by NOAA, US Air Force, NSF and NCAR through the Developmental Testbed Center (DTC)

NCAR

Developmental Testbed Center

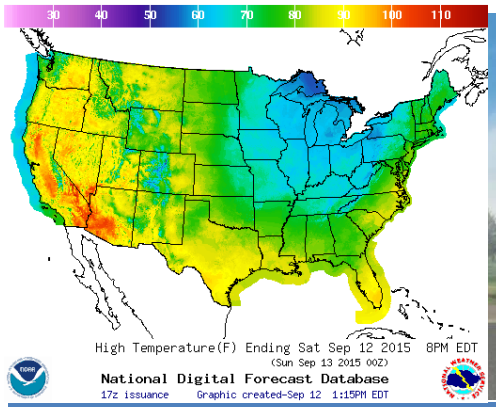
Unifying Verification through a Python-wrapped Suite of Tools

Tara Jensen, John Halley Gotway, Minna Win-Gildenmeister, Julie Prestopnik, Jim Frimmel, Geoge McCabe, Randy Bullock, Ivanka Stajner and Geoff Manikin

Workshop on developing Python frameworks for earth system sciences
ECMWF – Reading November 28-29

Why Unification

Forecasters



Government Centers

University and National Lab Researchers

Comprehensive and unified verification tool - Make R20 more efficient - Provide a consistent set of metrics

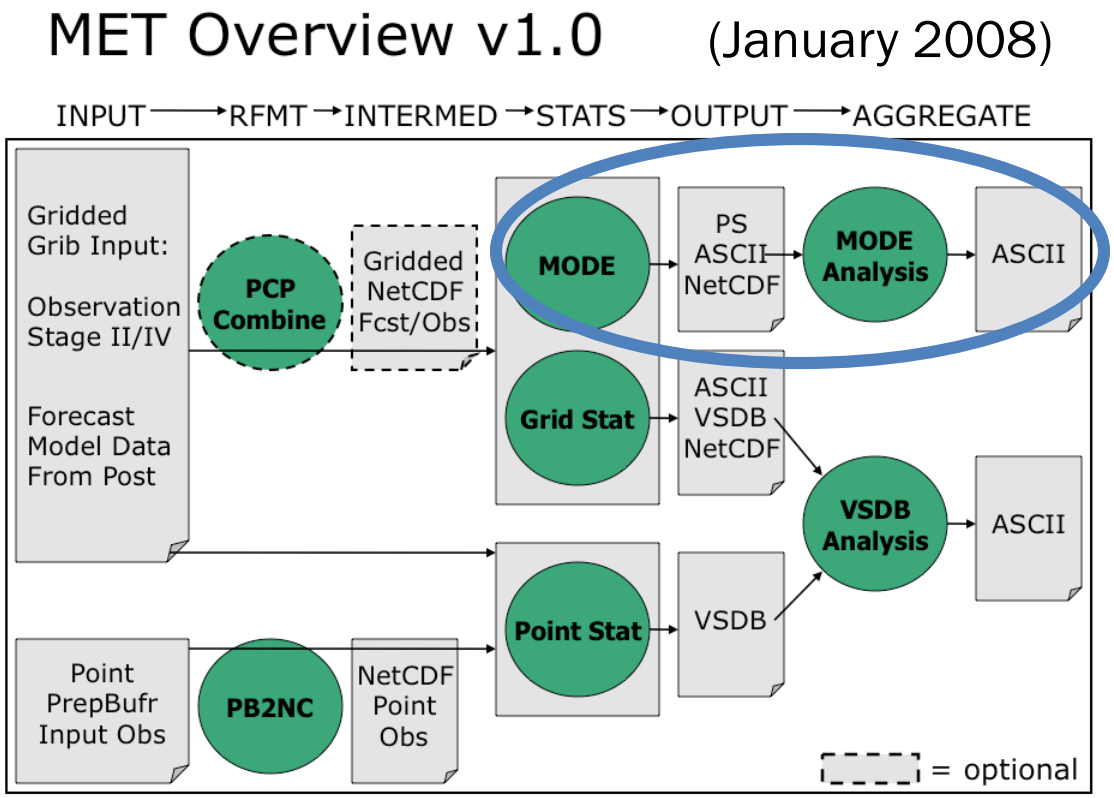
Allows Researchers and Operational Scientists to speak a "common verification" language

MET
Model Evaluation Tools

User Support of unified package provides greater opportunity to train all on verification best practices

A Bit of History - Over a Decade Ago

DTC was asked to replicate the EMC mesoscale verification package (VSDB) in a platform independent and extensible format and provide it to the community



Additional tools beyond EMC VSDB

MET Package - Today

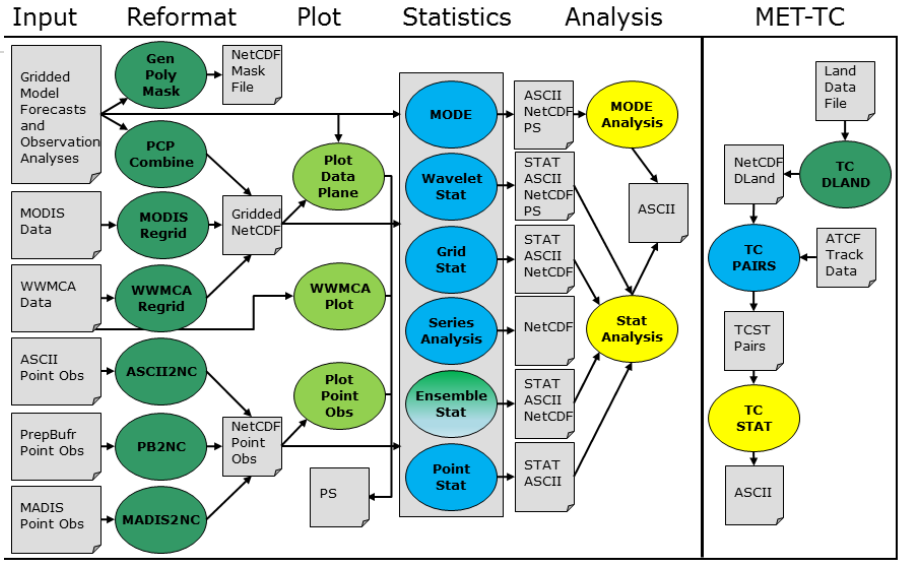
- MET is community code supported by DTC that is free to download (registration required)
  - 3400+ registered users
  - 125+ countries, 33% from USA
  - Universities, Government, Private Companies, Non-Profits
- Download MET release and compile locally.
  - Register and download: **www.dtcenter.org/met/users**
  - C++ with calls to some Fortran libraries, GSL, netCDF4 and HDF5
  - Linux with GNU, Portland Group (PGI), or Intel compilers
- Support
  - Online tutorial and in-person tutorials given yearly
  - **met_help@ucar.edu** help desk
  - 250+ support tickets in past year

Not adopted by EMC until recently Code-bases diverged

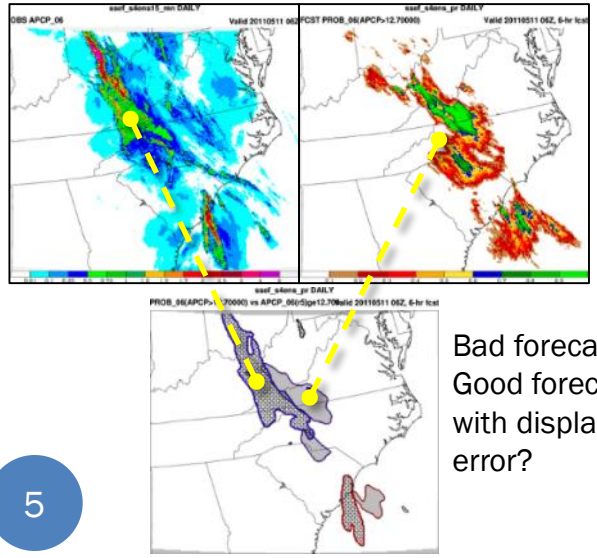
MET
Model Evaluation Tools

A verification toolkit designed for flexible yet systematic evaluation (supported to the community via the DTC)

- **Over 70 traditional statistics** using both point and gridded datasets
- Multiple interpolation methods
- Computation of confidence intervals
- Able to read in GRIB1, GRIB2 and CF-compliant NetCDF
- Applied to many spatial and temporal scales (multi-decadal climate to 15-min storm-scale)
- Regridding within the tools and ability to apply complex masking
- 3400+ users, both US & Int'l

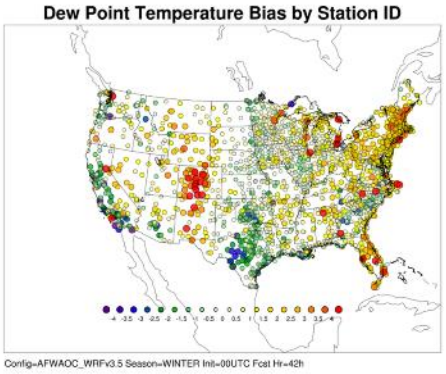


Object Based and Spatial Methods

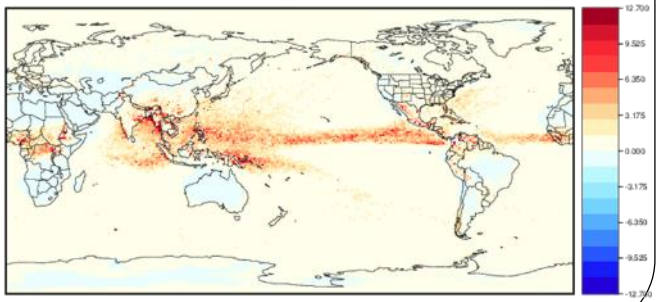


Bad forecast or Good forecast with displacement error?

Geographical Representation of Errors

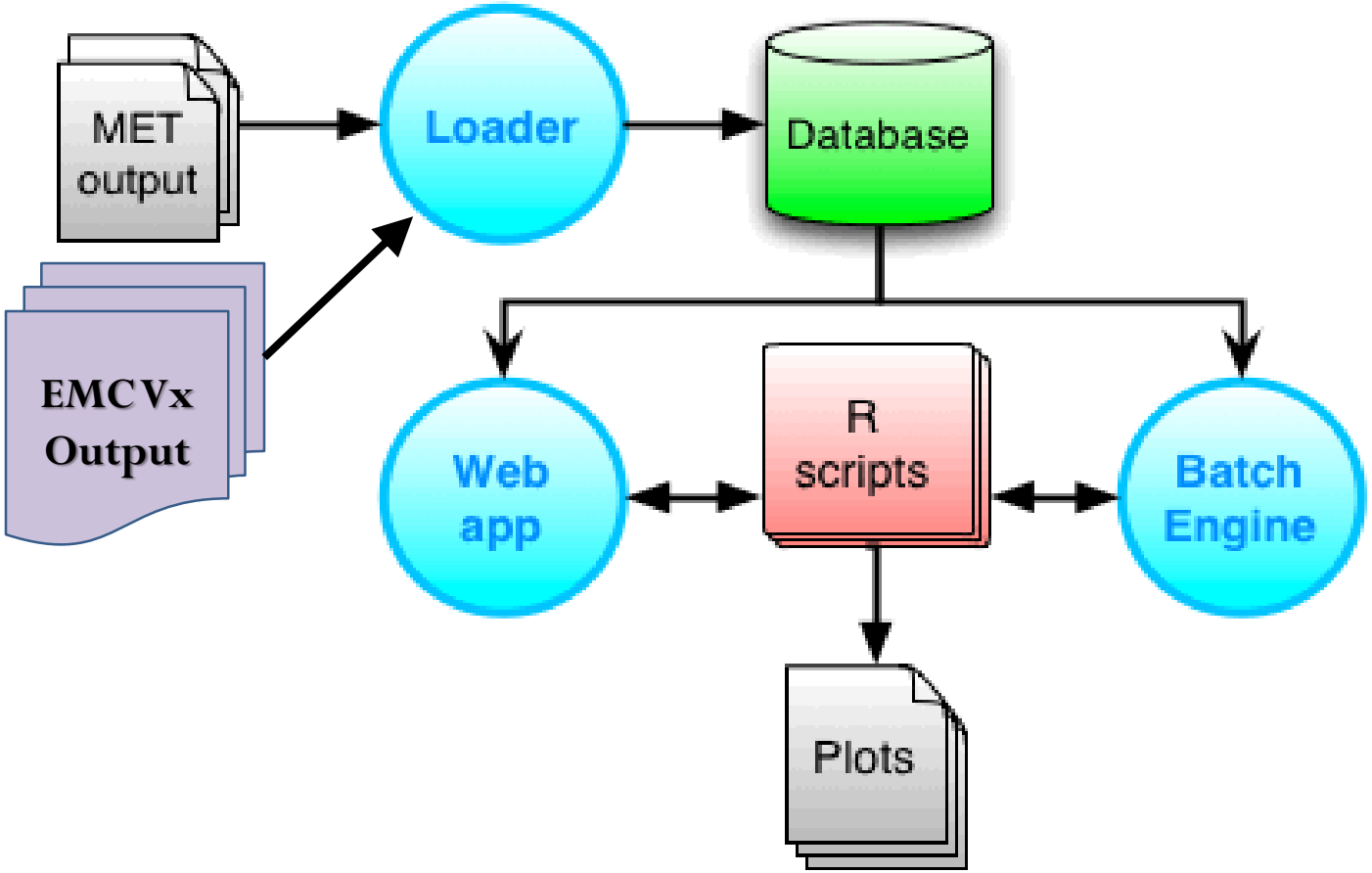


90th Percentile of difference between two models



METViewer components

Packages: Java, Apache/Tomcat, MySQL, R statistics

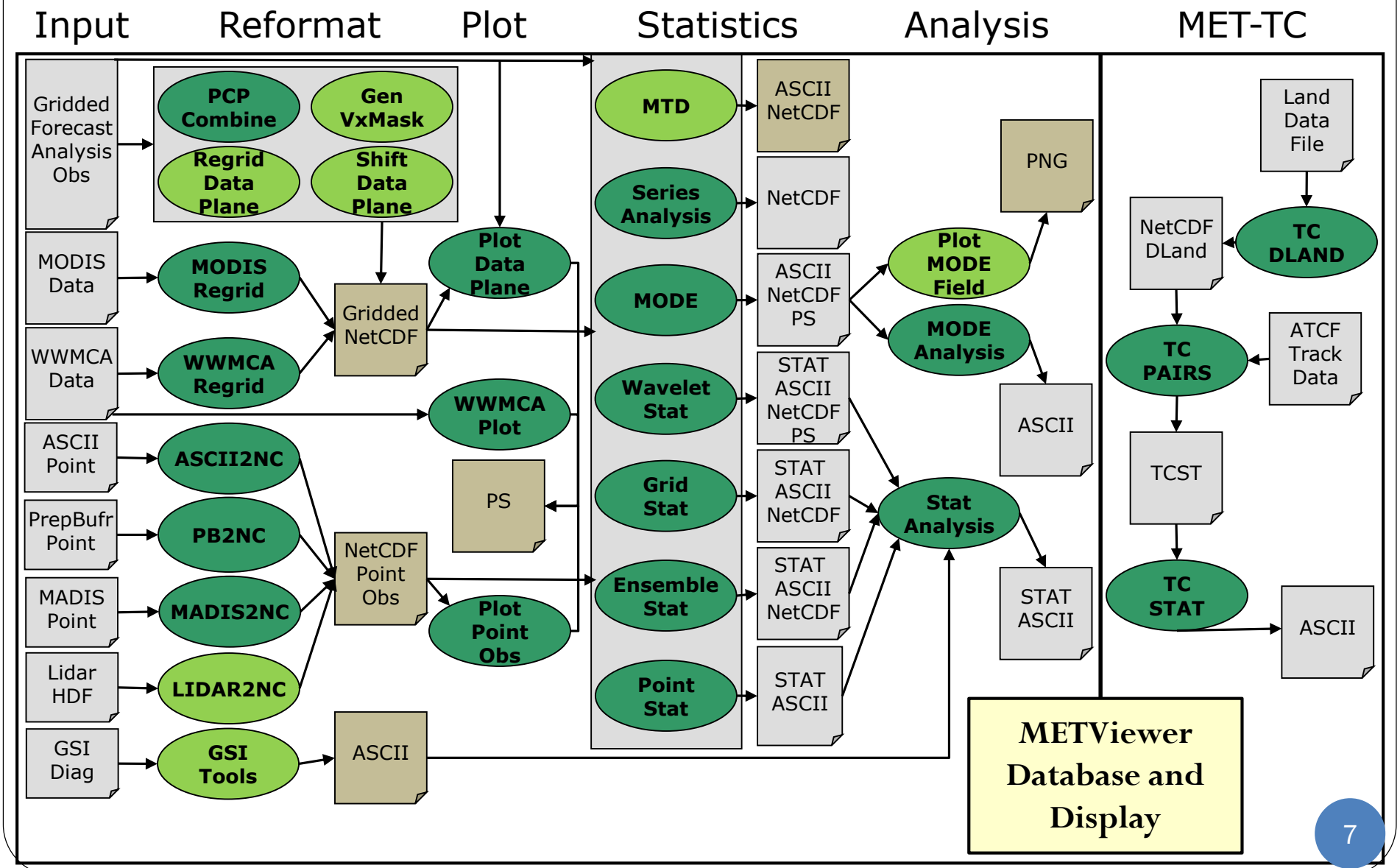


Database and Display analysis tool

MET v6.0 overview

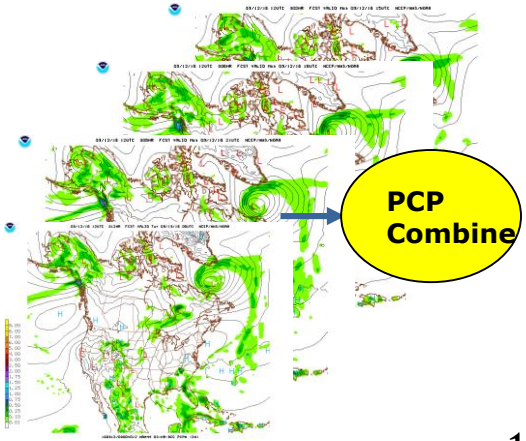
New tool in past 3 years

New output in past 3 years



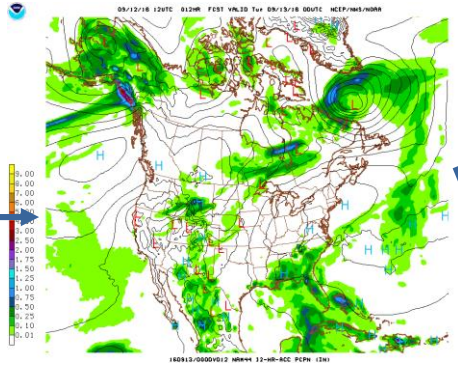
Example: Accumulated precipitation

3-h accumulation QPE

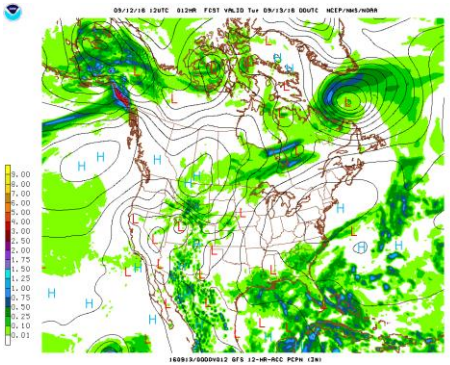


PCP Combine

12-h accumulation QPE



12-h accumulation QPF



MODE

MODE-TD

Wavelet Stat

Grid Stat

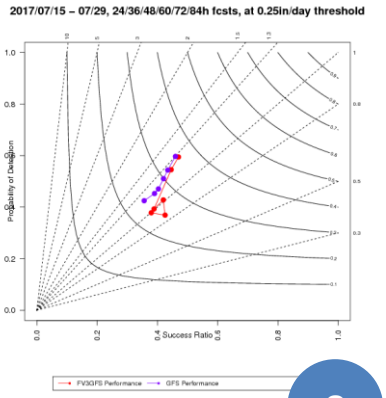
Series Analysis

Ensemble Stat

Point Stat

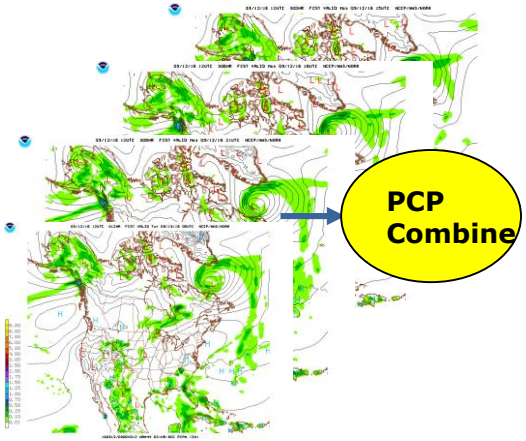
METViewer Database and Display

Multiple runs over time



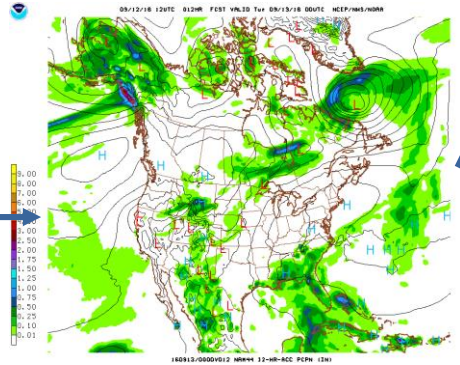
Example: Accumulated precipitation

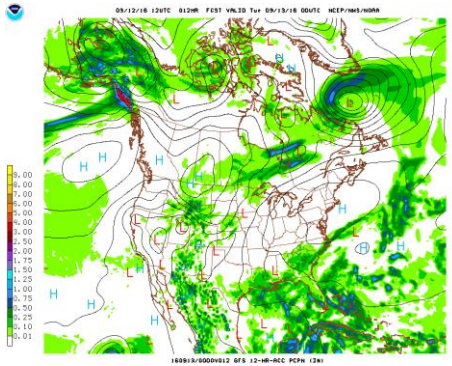
3-h accumulation QPE



PCP Combine

12-h accumulation QPE





12-h accumulation QPF

MODE

MODE-TD

Wavelet Stat

Grid Stat

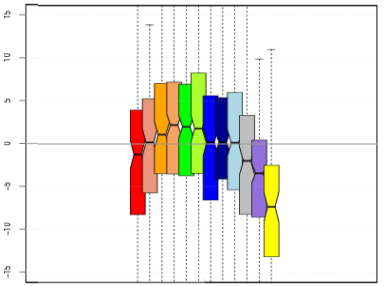
Series Analysis

Ensemble Stat

Point Stat

METViewer Database and Display

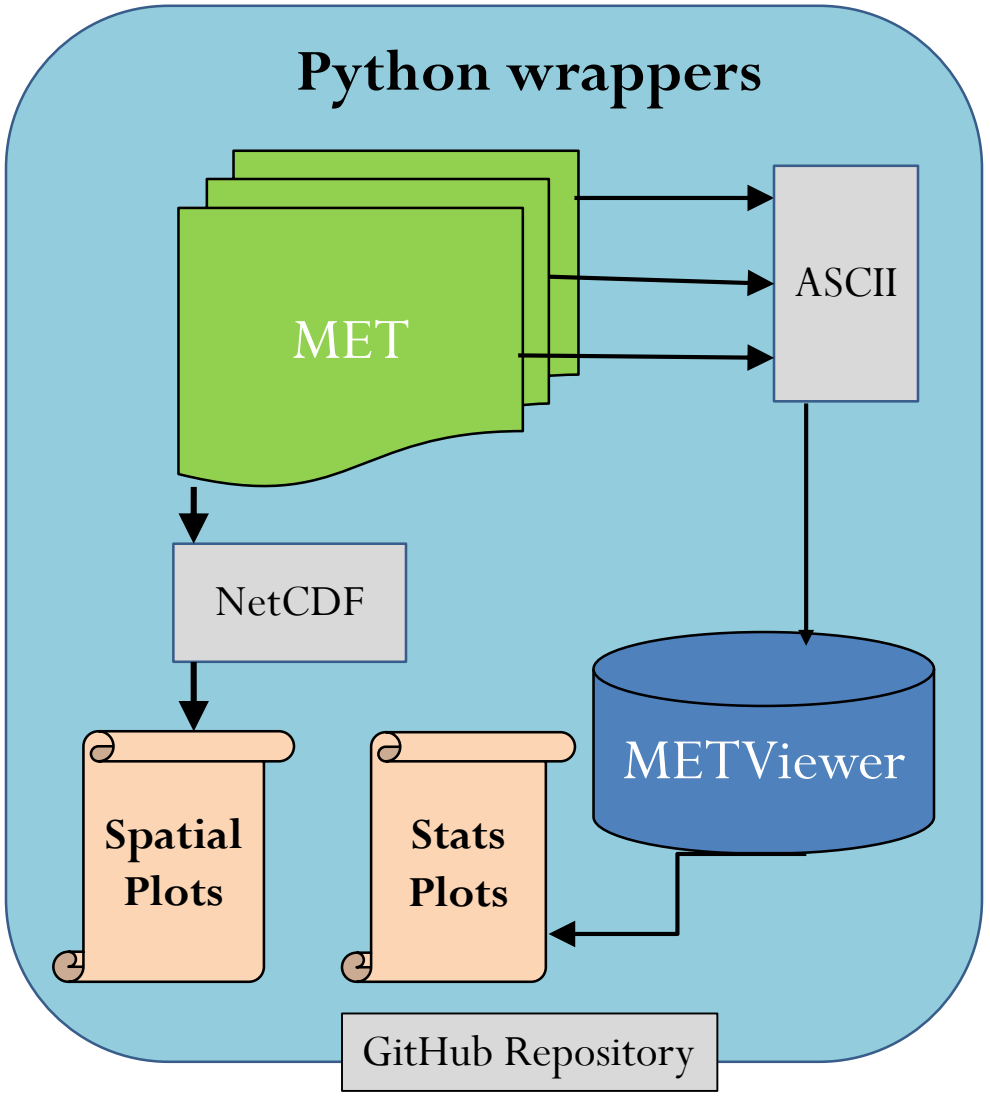
Multiple runs over time



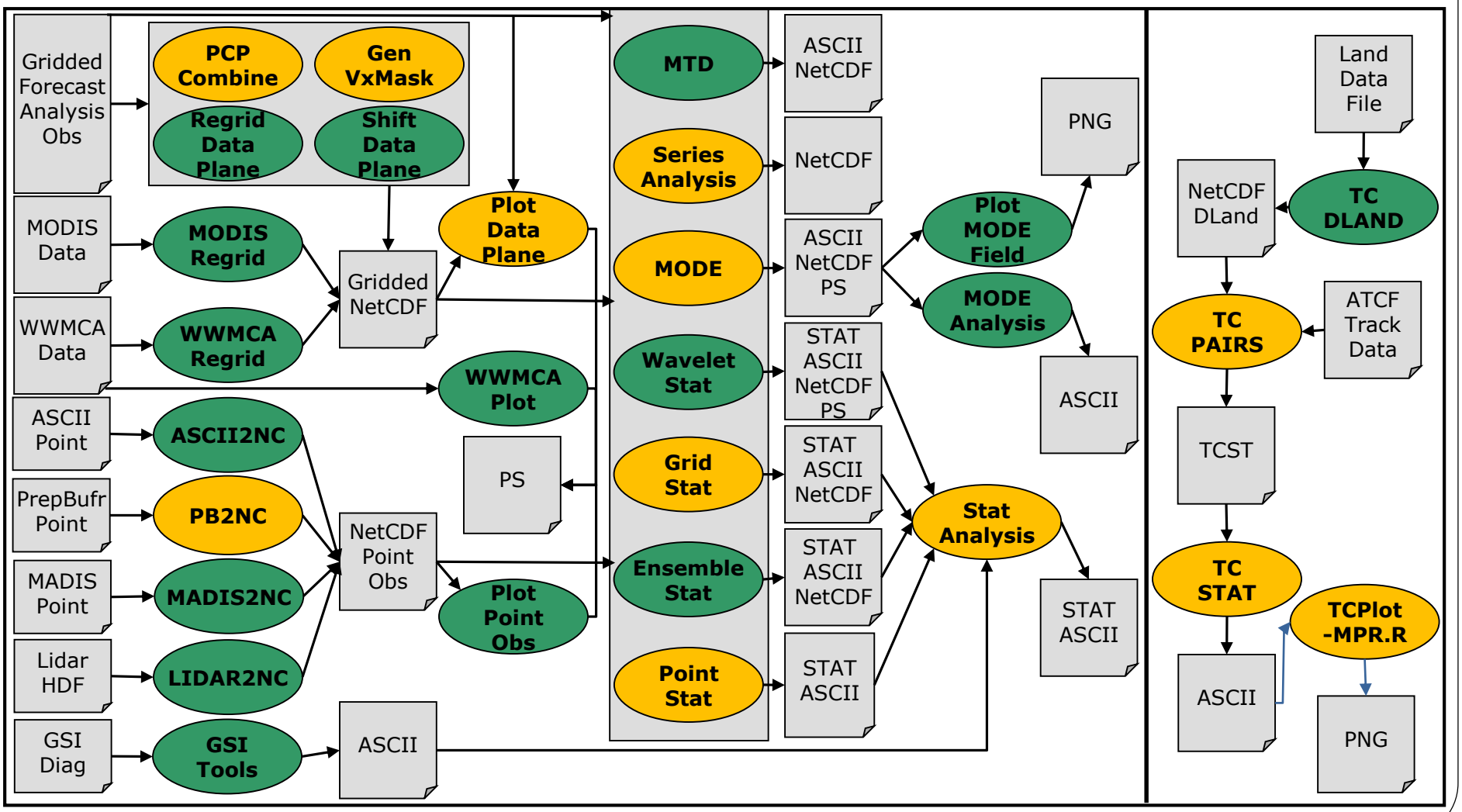
MET+ Unified Package

- Python wrappers around MET and METViewer:
- Simple to set-up and run
- Automated plotting of 2D fields and statistics
- Communication between MET & python algorithms (Cython)

Initial system - Global deterministic with plans to generalize across scales when possible to quickly spin-up Ensembles, High Resolution & Complete Earth System Model Components



By January



Copyright 2017, University Corporation for Atmospheric Research, all rights reserved

What does wrapped by Python mean?

MET/Series_Analysis_Config

In MET Configs: Environment variables passed in from MET+ Constants File

```
30
31  ////////////////////////////////////////////////////////////////////////////////
32
33  cat_thresh  = [ NA ];
34  cnt_thresh  = [ NA ];
35  cnt_logic   = UNION;
36
37  //
38  // Forecast and observation fields to be verified
39  //
40  fcst = {
41
42     field = [
43       {
44         name  = "${NAME}";
45         level = [ "${LEVEL}" ];
46       }
47     ];
48
49  }
50  obs = fcst;
51
52  ////////////////////////////////////////////////////////////////////////////////
53
54  //
55  // Climatology mean data
```

What does wrapped by Python mean?

METplus/parm/use_cases/feature_relative

feature_relative.conf

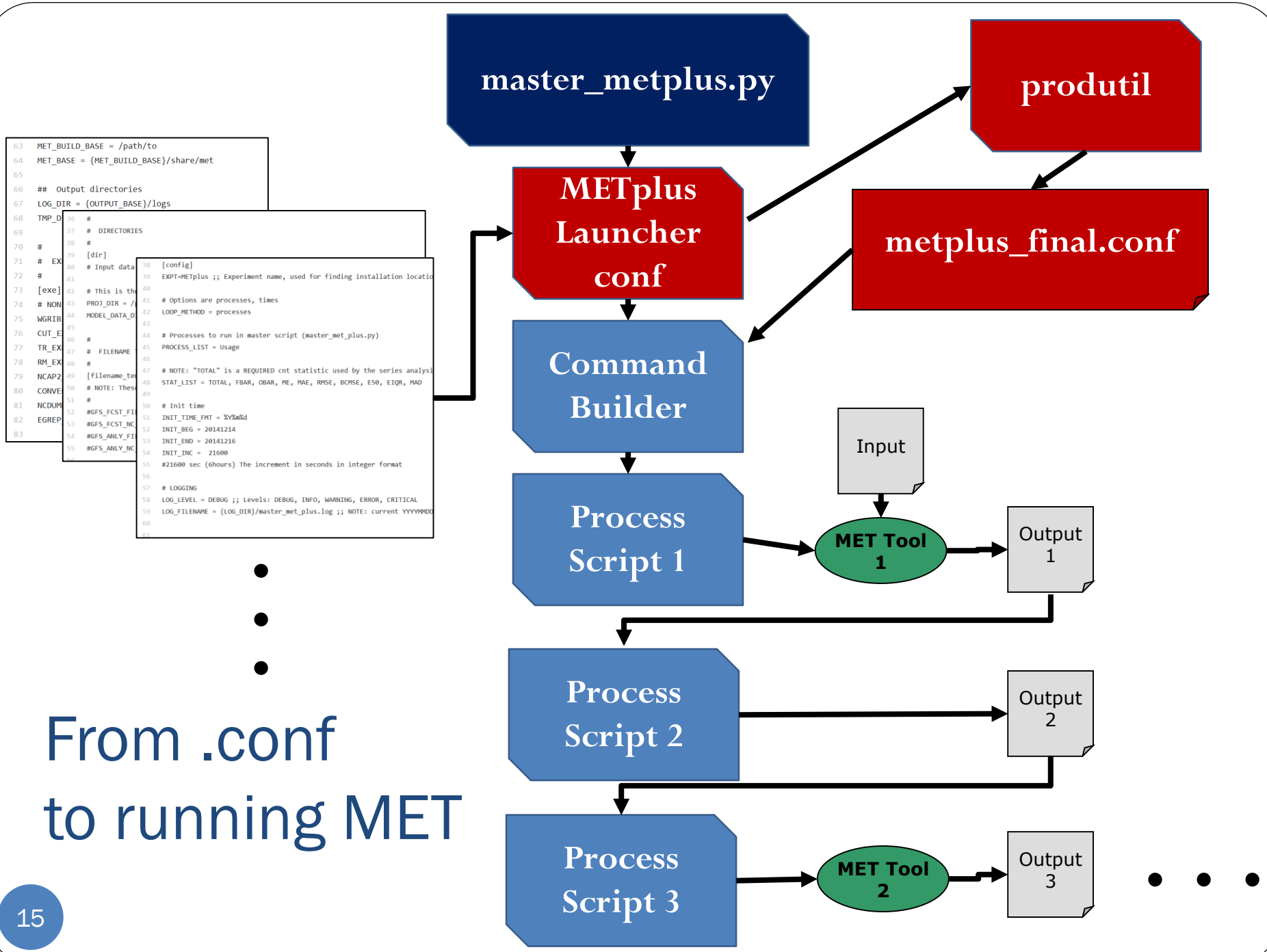
```
#
#     LISTS AND SETTINGS
#

#     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
```

What does wrapped by Python mean?

METplus/parm/use_cases/feature_relative

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
```


master_metplus.py
produtil
METplus Launcher conf
metplus_final.conf
Command Builder
Process Script 1
Process Script 2
Process Script 3
Input
MET Tool 1
MET Tool 2
Output 1
Output 2
Output 3
From .conf
to running MET

Current package dependencies

The Pyplot API

Matplotlib Basemap

MET+ Coding Standards

- NCEP Coding Standards
  - Python section of NCEP standards
  - Based on Google Python Coding standards
  - Language rules
  - version 2016a found on NCAR/METplus GitHub wiki:

https://github.com/NCAR/METplus

MET+ Coding Style

pep8 (Python Enhancement Proposal)

- code consistency and readability
- code layout: whitespace, line length, etc.
- naming conventions for variables, classes, methods and functions

https://www.python.org/dev/peps/pep-0008

MET+ Documentation

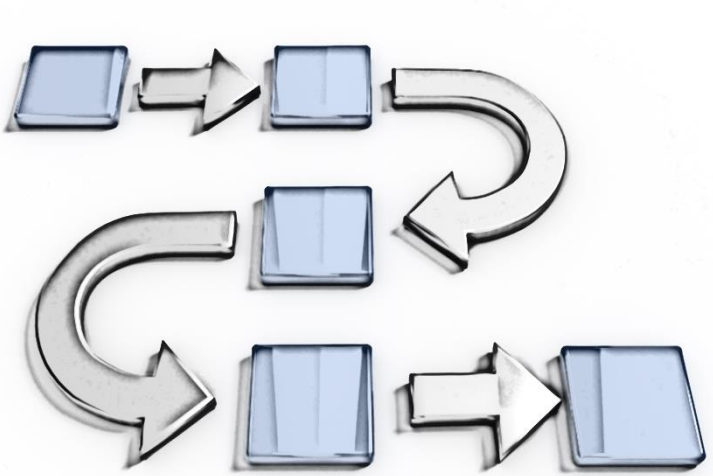
- Doxygen
  - documentation generator
  - written in the source code
  - easy to keep up-to-date
  - documentation can be readily viewed (html documentation generated)
- Python docstrings for information about functions, methods
  - help(python object)
  - readily generated from Python REPL (language shell)
  - REPL = read, evaluate, print, loop

MET+ Best Practices

- Run pep8 tool or similar application to verify pep8 conformance
- Run pylint or other code linters to ensure code readability and to identify unused variables, imports, etc.
- Run relevant tests written in Python unittest
- Use GitHub for code management including issue tracking and wiki

Aligning with NCEP Workflow

- Discussing and planning to collaborate on new dynamical core (FV3) workflow developers.
- Using PRODUTIL package for logging and constants file parsin plus likely other features
- Plan to use Rocoto workflow management (NOAA tool) for dev environments and make autonomous to also use with ecFlow or Cylc for operations



MET+ Beta - Prerequisites

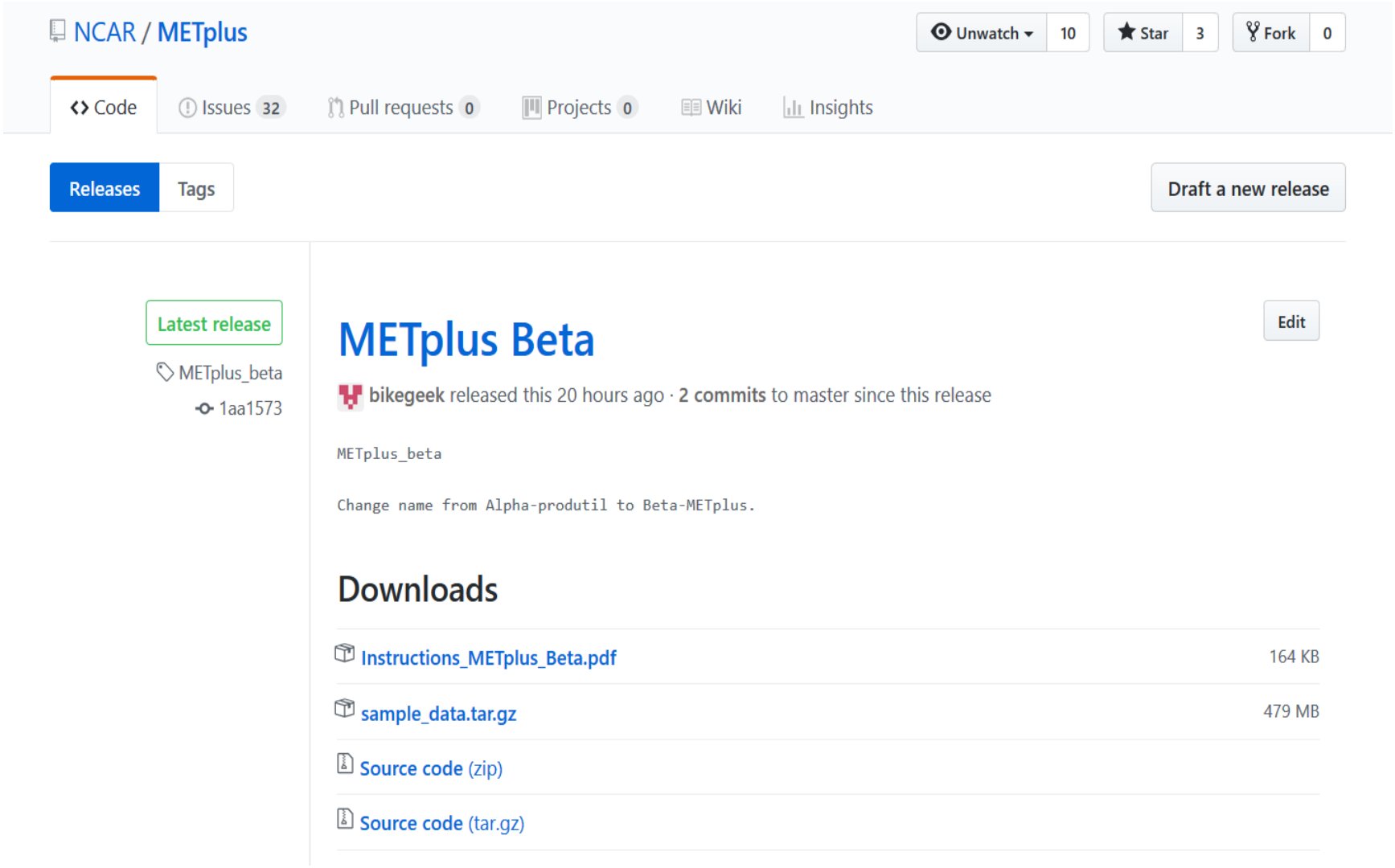
- Python 2.7 ***When we started this was specified by NCO*
- R version 3.25 ** *Only if you are using PlotTCMpr.R tool*
- nco (netCDF operators)
- MET version 6.0 or later installed
  ** *Tool is designed to sit on-top of MET and should be version insensitive after METv6.0*
- Basic familiarity with MET

- **User:** Access the public release at: https://github.com/NCAR/METplus/releases

-OR-

- Use install on Theia or WCOSS
  ** *Only on Gyre right now, will populate on Surge, Tide and Luna as access is available*
- **Developer:** Need a github account https://github.com/NCAR/METPlus/ then proceed like a User

Grabbing the Release



Operational Directory Structure

- doc/ - Doxygen documentation
- internal_tests/ - developer tests
- parm/ - where configs live
- README.md - general README
- sorc/ - executables
- ush/ - python scripts

Suggestions on how to set up parm dir

- **Met_config**
  - All MET configuration files with Environment Variables should reside here
- **Metplus_config**
  - Common install for COMMUNITY INSTALLATION – includes paths to commonly used data
- **Use_cases**
  - Common install for FUNCTIONAL GROUP – includes paths for tests your conducting
- **{user}_system.conf.{system_name}**
  - Place your variances from use-cases in here, including pointing to your output directory, or pointing to a different config you are trying, etc…

3 Use Cases

- Cyclone Track and Intensity
  - Using MET-TC to pair up ATCF track files
  - PlotTCMPR.R to compute track and intensity errors and plot

- Feature Relative
  - Use MET to match up lat/lon pairs of forecast and analysis feature (e.g. TC, Extra-tropical cyclone, snowbands, MCS, jet streak, etc…)
  - Extract user specified tiles from Forecast and Analysis files for computation of systematic errors
  - Use Series-Analysis to compute statistics for the stack of tiles over region of interest (e.g. Eastern US, UK, Central Europe, Middle East, Tropics)
  - Use Plot-Data-Plane to generate quick look plots

- QPF
  - Use Pcp-Combine to accumulate 1-hr QPE into 3-hr accumulation
  - Use Grid-Stat to compute Categorical statistics (e.g. Prob of Detection, False Alarm Ratio) for multiple thresholds

Next Major Development: Communication between MET and Python

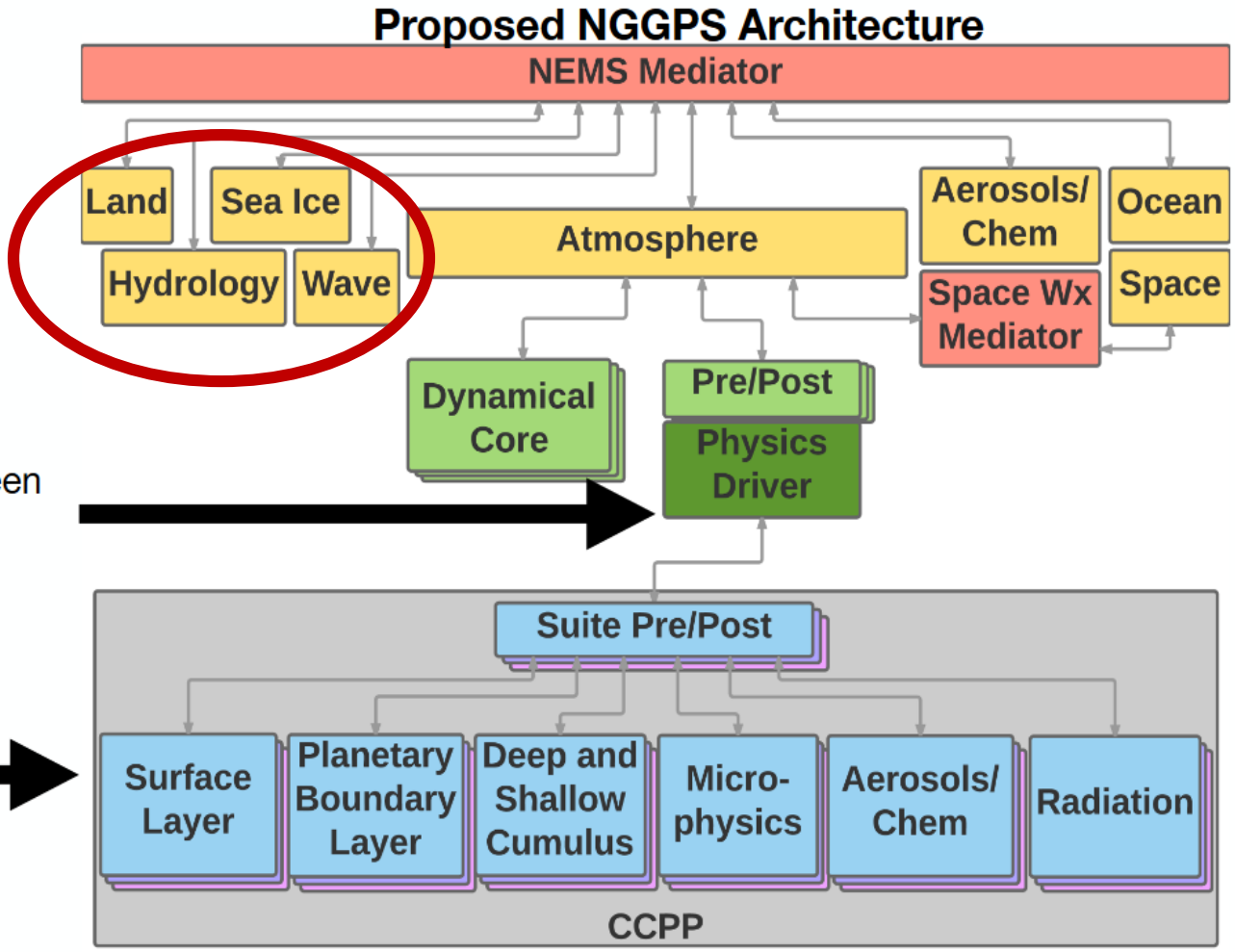
Anticipated Solution: Cython - C extentions for python

- Write Python code that calls back and forth from and to C or C++ code natively at any point.
- Easily tune readable Python code into plain C performance by adding static type declarations.
- Use combined source code level debugging to find bugs in your Python, Cython and C code.
- Integrate natively with existing code and data from legacy, low-level or high-performance libraries and applications.

Excerpted from: http://cython.org/

Next… Adding Additional Capability

Already have pre-existing packages which we will link in



IPD: Common interface between multiple dycores and physics suites

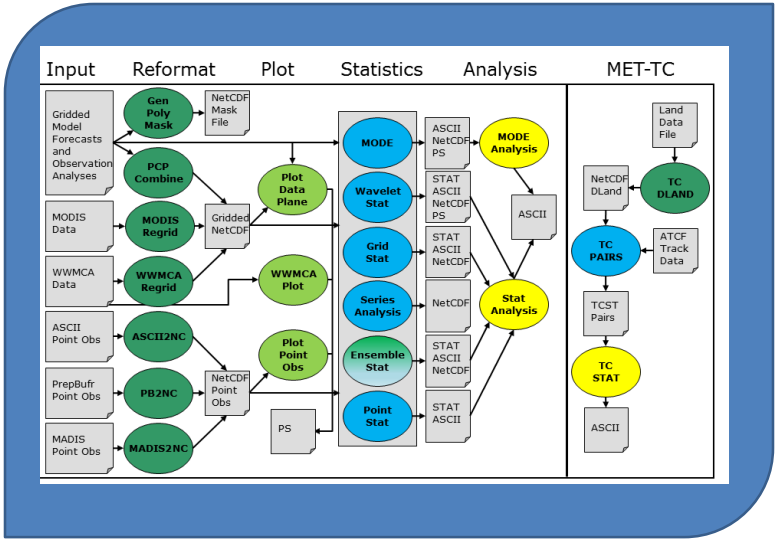
CCPP: Limited collection of physics suites that interface with the IPD

Next year will be adding more process-oriented diagnostics to aid in CCPP development

Docker MET and METViewer

Docker (Amazon Web Services):

- Open-source technology to build and deploy applications inside software containers
- Packages software containing: code, runtime, system tools, system libraries, etc
- Enables you to quickly, reliably, and consistently deploy applications



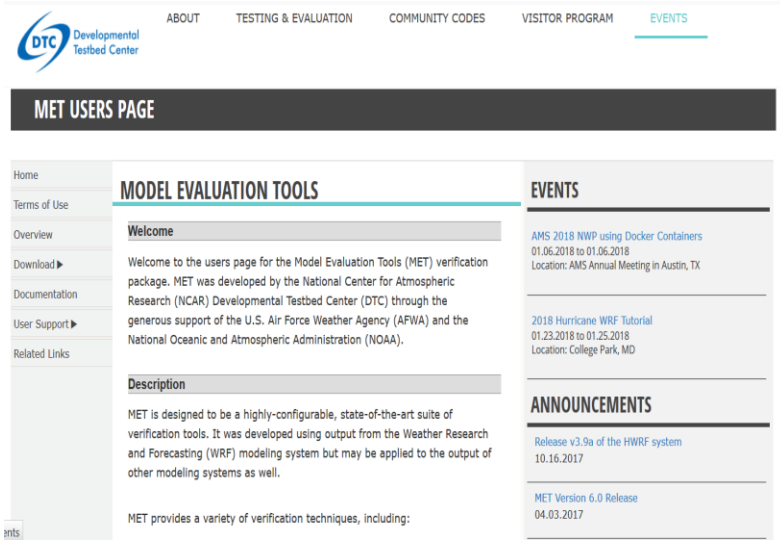
MET and METViewer compiled in a Docker Container

1) Set up to work with a suite of test-cases for NWP innovation testing (MMET/MERIT)
2) Bundled with MET online tutorial data

http://www.dtcenter.org/met/users/downloads/docker_container.php

Thank You

Where to find MET+ and get help



- GitHub Instructions at Release link
  - https://github.com/NCAR/METplus/releases
  - Instructions_METplus_Beta.pdf
- MET Users Page with documentation and instructions for obtaining Docker MET and METViewer: https://dtcenter.org/met/users
- Contact met_help@ucar.edu