# **MUSICA Tutorial Series 2021-2022:** Introduction to MELODIES-MONET

**MUSICA: MUlti-Scale Infrastructure for Chemistry and Aerosols** 



11 March 2022



**MELODIES:** Model EvaLuation using Observations, Dlagnostics and Experiments Software

**MONET**: Model and ObservatioN Evaluation Toolkit

Current developers:

Rebecca Buchholz, David Fillmore, Duseong Jo, Louisa Emmons - NCAR/ACOM

Becky Schwantes, Barry Baker, Zachary Moon, Maggie Bruckner, Meng Li, Jian He, et al. – NOAA (CSL, GSL, ARL)

A modular framework to compare model results and observations of atmospheric chemistry
In Python, using Xarray, NumPy, SciPy, Pandas, ...
Goal: to be able to read any model and pair it to observations (surface in situ and remote sensing, aircraft, satellite, ...)





Warning: Still under development!



MELODIES is built on MONET & MONET-I/O: Model and ObservatioN Evaluation Toolkit https://monet-arl.readthedocs.io/en/master/

MONET I/O

.

Model Object Preparation

and variables

longitudes

Reads global attributes, grid

Converts time, latitudes and

#### MELODIES-MONET

Configuration file

- Define comparisons and analysis
- Call driver script
- Python notebook workflow
- Development scripts

#### MONET I/O

Measurement Object preparation

- Retrieves data if necessary and possible (url, OpenDap, Amazon cloud)
- Converts to netCDF if necessary
- Reads and exports to dataframe

Written in Python Using standard packages: Xarray, NumPy, SciPy, Pandas, ...

Jupyter Notebooks with examples



Pairing and Processing Objects

- Pairs observations and model
- Methods for visualization and statistics

Goal: Output that can be browsed in a html page format User's Guide provides instructions for adding new models or observations

#### MELODIES-MONET



#### Analysis Class Time window Model Class **Observation Class** Model label and type Reads in yaml file Obs type: surface, aircraft, & satellite ٠ Mapping dictionary for pairing model • Pairing class Unit conversions for each variable . and observation variables All models are paired including handling NaNs, Min, and Max Radius of influence • with observations Axis scales for each variables used for • Color/markers for plots • specified by mapping plotting Unit conversions if needed dictionary Plotting Statistics For each plotting group: List of statistics to calculate • Plot type List of model/obs pairs to loop over . . Automatically loop over all variables specified in the List of domains to loop over • . model class Outputs a .csv file and an optional table graphic . List of model/obs pairs to loop over for each variable specified in the model class and ٠ List of domains to loop over for each model domain listed. ٠ Default plotting info for figure, text, & maps properties . Data processing specifications (e.g., NaN handling, ٠ timeseries averaging, axis range)

Fig. 1: Schematic of the classes defined in the Python code and used by MELODIES MONET.



# **MELODIES-MONET Driver:**

- 1. Reads in the yaml file
- 2. Reads in the model results
- 3. Reads in the observations
- 4. Pairs the model and observations and stores them as class "pair"
- 5. Then the plotting routines reads this class "pair" and calls general plotting routines to make the plots, with options such as:

Calculate regulatory metrics (PM2.5 daily average, and MDA8 ozone) Separate by region (depending on categories available in dataset)

For each of the observations, unit conversions will be applied.

- Gases are all in ppbv and aerosols in ug/m3
- others like meteorology are not as standardized yet

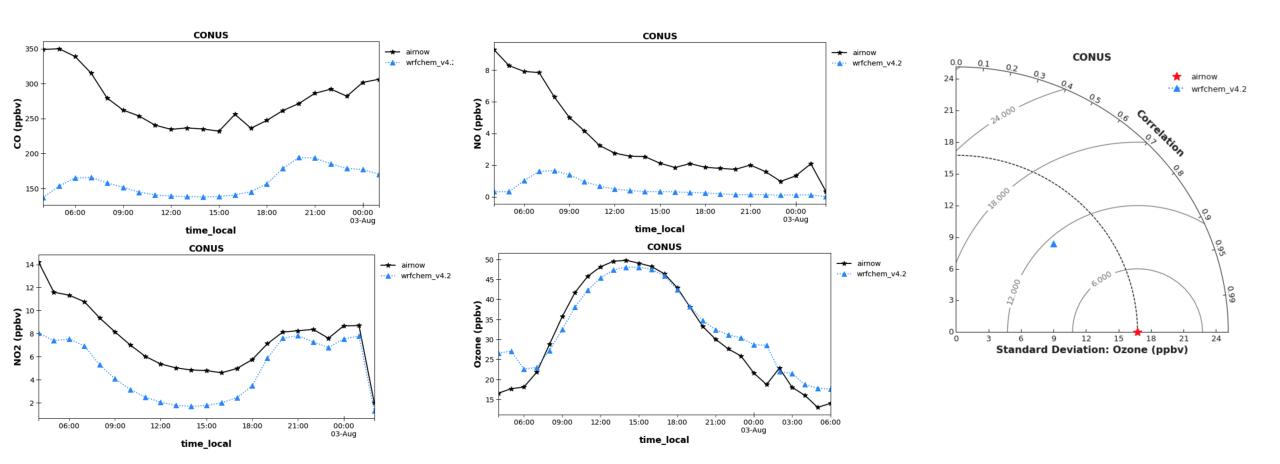
MELODIES-MONET: <u>https://github.com/NOAA-CSL/MELODIES-MONET</u> Documentation: <u>https://melodies-monet.readthedocs.io/</u> #This just pairs the data
an.pair\_data()

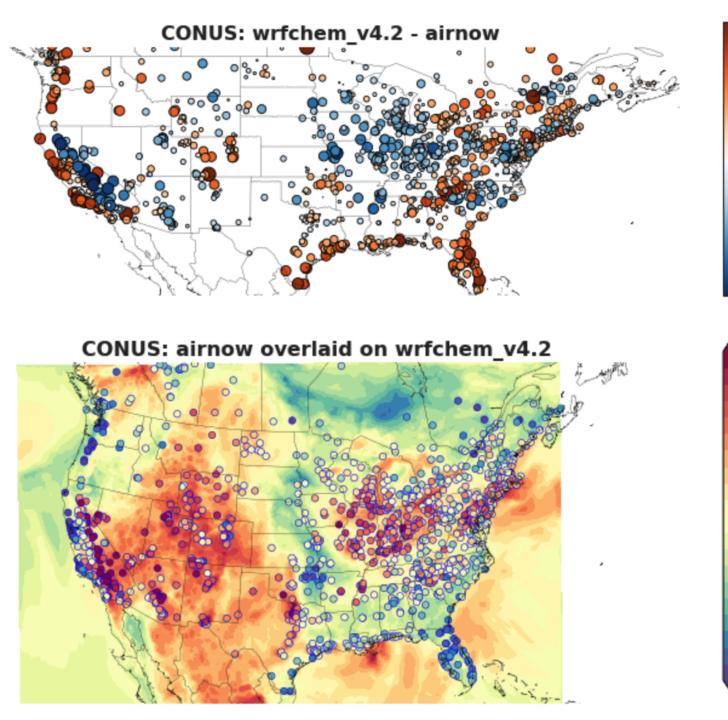


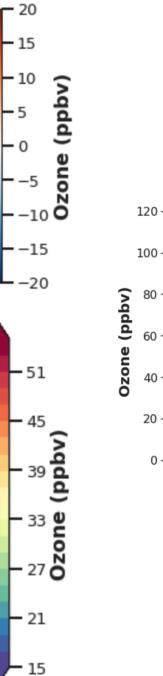
[######################################		100%	Completed		0.1s
[######################################	Ì	100%	Completed	ĺ	0.1s
[######################################		100%	Completed		0.2s
[######################################		100%	Completed		0.2s

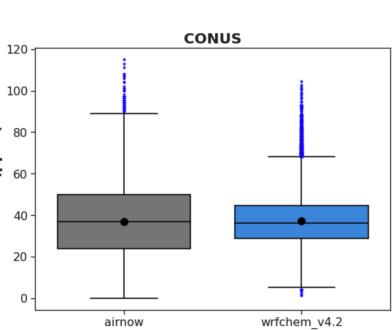
#### In [12]:

### #And this generates all the plots. an.plotting()













# **Examples of Output: Statistics**

# August 2019 - Average in WRF-chem

Statistics of OZONE and o3 pair over file period 2019-08-01\_06 to 2019-09-01\_05

Number of OZONE Observations = 39986 Number of o3 Predictions = 39986 Number of OZONE/o3 Observations/Prediction Pairs (#) = 39986 Mean of OZONE Observations = 42.82Mean of o3 Predictions = 46.30Median of OZONE Observations = 42.50Median of o3 Predictions = 44.86Standard deviation of OZONE Observations = 12.91 Standard deviation of o3 Predictions = 11.43 Mean Bias of o3-OZONE = 3.48 Normalized Mean Bias (%) of o3-OZONE = 8.13 Normalized Mean Error (%) of o3-OZONE = 16.74 Root Mean Square Error of o3-OZONE = 9.26 Index of Agreement of o3-OZONE = 0.85 Pearsons Correlation Coefficient of o3-OZONE = 0.76

Statistics of PM2.5 and PM25 pair over file period 2019-08-01\_06 to 2019-09-01\_05

Number of PM2.5 Observations = 27981 Number of PM25 Predictions = 27981 Number of PM2.5/PM25 Observations/Prediction Pairs (#) = 27981 Mean of PM2.5 Observations = 7.06Mean of PM25 Predictions = 9.87 Median of PM2.5 Observations = 6.50 Median of PM25 Predictions = 8.09 Standard deviation of PM2.5 Observations = 3.25 Standard deviation of PM25 Predictions = 6.43 Mean Bias of PM25-PM2.5 = 2.82Normalized Mean Bias (%) of PM25-PM2.5 = 39.91 Normalized Mean Error (%) of PM25-PM2.5 = 67.45 Root Mean Square Error of PM25-PM2.5 = 6.63 Index of Agreement of PM25-PM2.5 = 0.49 Pearsons Correlation Coefficient of PM25-PM2.5 = 0.38 



# Development plans

### Short-term, in progress:

- Reading aircraft campaign observations
- Reading satellite retrievals, applying averaging kernels, for swaths & gridded
- Reading additional models

## Longer-term goals:

- MELODIES-Climate same structure as MELODIES-MONET but would compare model(s) to climatologies of observations
- Connect with METPLUS
  - Have METPlus call MELODIES-MONET
  - Take advantage of MET statistics and plotting
  - METplus will use MONET I/O



# Please get involved and contribute!

MELODIES MONET is a community-driven project.

We welcome collaborations and contributions.

- Ask questions, suggest features, or view source code on GitHub: <u>https://github.com/NOAA-CSL/MELODIES-MONET</u>
- If an issue arises, please post on GitHub Issues: <u>https://github.com/NOAA-CSL/MELODIES-MONET/issues</u>
- Please check out GitHub Projects: <u>https://github.com/NOAA-CSL/MELODIES-MONET/projects</u> to learn about current development plans

# Installation of MELODIES-MONET

Please find instructions specific for this tutorial in this doc:

https://bit.ly/melodies-tutorial

General installation instructions are in the MELODIES-MONET documentation: <u>https://melodies-monet.readthedocs.io/en/develop/tutorial/installation.html</u>

As the code is currently in a state of rapid development, please check the melodies-monet site for the latest instructions

# Hands-on examples

- Minimal example (WRF-Chem vs AIRNOW) David Fillmore (NCAR)
- Idealized example Zachary Moon (NOAA)
- Using MUSICA (variable unstructured model grid) Duseong Jo (NCAR)
- Comparing multiple models (Taylor diagram) Rebecca Buchholz (NCAR)

### End of presentation - switch to JupyterHub notebooks

The following slide provide some instructions and tips for getting started on NCAR's JupyterHub system and starting the example notebooks.

**1. Start NCAR JupyterHub**: <u>https://jupyterhub.hpc.ucar.edu/</u> (you must have an account on cheyenne/casper)

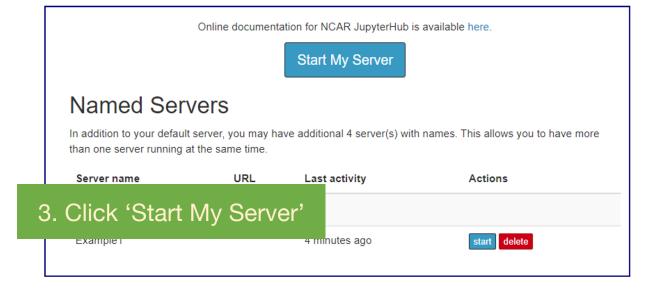
If you need more information see:

https://arc.ucar.edu/knowledge\_base/70549913#JupyterHubatNCAR-Gettingstarted

2. Select Production

### 3. Login and start server





#### Select Casper PBS batch (or Cheyenne PBS batch)

- You need to use casper to access campaign storage
- Using a batch node allows you to specify project number and memory

### NCAR HPC JupyterHub



Casper login node

Casper login node

Casper PBS batch

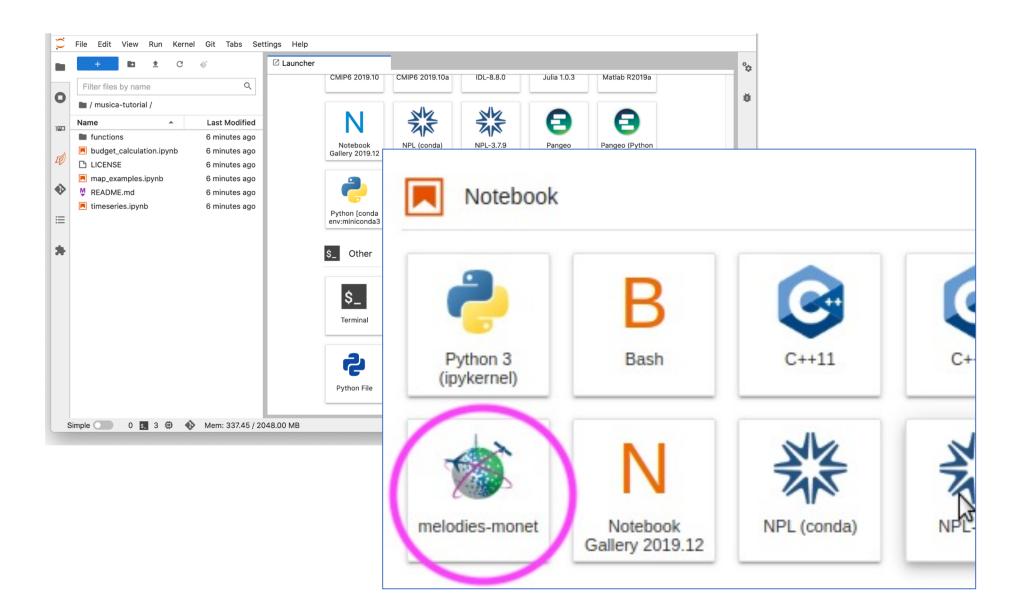
Chevenne login node Chevenne PBS batch

NCAR HPC JupyterHub **Cluster Selection** Casper PBS batch Enter Queue or Reservation (-q) casper -Enter your project number Specify your project account (-A) Specify N node(s) (-I select=N) You may need to increase memory, e.g. 15 GB Specify N CPUs per node (-I ncpus=N) Specify N MPI tasks per node (-I mpiprocs=N) Increase wall-time only if needed Specify N threads per process (-I ompthreads=N) Specify the Amount of memory / node in GB (MAX: 1494) Specify X Number of GPUs / Node (-I ngpus=X) To reduce waiting time for Hub to start, 0 use smallest memory and wall-time Select GPU Type, X (-I gpu\_type=X) none possible Specify wall time (-I walltime=[[HH:]MM:]SS) (24 Hr Maximum) 02:00:00 Launch Server

If the computer is busy you may need to wait for the server to connect....

NCAR	Home	Token	emmons	C Logout
			Your server is starting up. You will be redirected automatically when it's ready for you.	
		Event log	Cluster job running waiting to connect	

In the Launcher window look for the 'melodies-monet' kernel. Having this means you will also be able to select it from within your notebooks.



*If* you do not have access to the 'melodies-monet' kernel, you will need to create your own MELODIES MONET conda environment.

Follow the instructions at:

https://melodies-monet.readthedocs.io/en/latest/appendix/machine-specific-install.html#ncar-hpc-cheyenne-casper

You may wish to follow these instructions if you plan to develop new code and contribute it to the MELODIES-MONET repository.

**If you have made your own melodies-monet conda environment on cheyenne/casper:** you should see a kernel under the JupyterHub Notebooks "conda-melodies-monet" that you should use when running the MELODIES MONET notebooks.

If you prefer to install MELODIES MONET on your own computer, see: <u>https://melodies-monet.readthedocs.io/en/latest/tutorial/installation.html</u>

Note: the instructions include 'git clone' commands using ssh, which assumes you have set up an ssh RSA key between cheyenne and your github account.
If you do not have that, you can use commands like:
> git clone https://github.com/NOAA-CSL/MELODIES-MONET.git
[instead of git clone git@github.com:NOAA-CSL/MELODIES-MONET.git]

#### **Tutorial example notebooks:**

- On cheyenne/casper: /glade/p/acom/MUSICA/melodies\_tutorial/notebooks/
- <u>https://drive.google.com/drive/folders/1-7NIQaUDlesGmGDU1P4VmlVII4P68rdI?usp=sharing</u>

Copy the \*.ipynb and \*.yaml files in either of these locations to somewhere in your home directory on casper (for example /glade/u/home/<username>/melodies\_tutorial/; you can open a terminal from the JupyterHub launcher to do this). Navigate within the JupyterHub file system display to that location and open a python notebook by double clicking (start with the mm basics.ipynb).

Be sure to select the 'melodies-monet' kernel after you open the notebook.

Before trying to run the notebook, open the .yaml file that the notebook reads (e.g., mm\_basics.yaml – it should be in the same directory as your notebook). Check that the paths for output, model results, observations are appropriate.

In the notebook, you may need to provide the full path to the yaml file.

#### When running tutorial notebooks:

Warnings related to "FutureWarning" after running "from melodies\_monet import driver" can be ignored. from melodies\_monet import driver

/glade/u/home/cdswk/python/miniconda3/lib/python3.7/site-packages/dask/dataframe/utils.py:15: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead. import pandas.util.testing as tm Please install s3fs if retrieving from the Amazon S3 Servers. Otherwise continue with local data Please install h5py to open files from the Amazon S3 servers. Please install h5netcdf to open files from the Amazon S3 servers.

#### or if you don't want to see warning messages:

```
# To ignore warning messages
import warnings
warnings.filterwarnings('ignore')
from melodies_monet import driver
Please install s3fs if retrieving from the Amazon S3 Servers. Otherwise continue with local data
Please install h5py to open files from the Amazon S3 servers.
Please install h5netcdf to open files from the Amazon S3 servers.
```