Last Updated: 5/5/2025

If in doubt, start by checking out the docs in https://melodies-monet.readthedocs.io/en/stable/

Quick Start Guide: MELODIES-MONET

The recommended operating system to use MELODIES-MONET is Linux. It will work on MacOS machines, but, depending on the hardware, you might encounter minor issues if you are reading WRF-Chem output. If you are using a MacOS computer with Apple Silicon CPUs and require WRF-Chem output, you will need to compile wrf-python from source (https://wrf-python.readthedocs.io/en/latest/installation.html). This might require installing precise versions of NumPy and other dependencies. More information is provided below.
If you are using Windows, it is recommended to use the Windows Subsystem for Linux, which is provided free of charge by Microsoft. Check out https://learn.microsoft.com/en-us/windows/wsl/install. This requirement could be especially important if you are using MELODIES-MONET to compare models with satellite output.

If you have any issues installing or running MELODIES MONET please contact Pablo Lichtig (plichtig@ucar.edu) or Gabriele Pfister (pfister@ucar.edu).

Prerequisites

This tutorial assumes that you have some basic familiarity with using the command line. If you don't, there are multiple tutorials freely available online, and they are usually sufficient . You can check out https://www.twilio.com/docs/usage/tutorials/a-beginners-guide-to-the-command-line for a general overview, or search "command line crash course" for your specific operating system.

Installation

The suggested installation method uses conda. If you are working on an HPC system, you probably already have it installed. Something similar to as used for the NSF NCAR HPC

```
module load conda/latest
```

should do the trick. Adapt it to your specific system (for instance, you might not need the `/latest`, or might need to set up your specific version, which you can check with `module avail`).
If you don't have it installed check this link for install instructions https://docs.anaconda.com/miniconda/install/.

Once you have conda installed, the minimum requirements for running MELODIES-MONET will be satisfied by simply running (don't rush to type it! Read until the next command instructions):

```
conda create --name melodies-monet -y -c conda-forge python=3.11
melodies-monet
```

However, for a more user-friendly handling it is recommended that you also install `jupyter-lab`. If you intend to read WRF-Chem output, you will also need to install `wrf-python`[1]. Install also `regionmask` to have extra region selection capabilities (which allows to select countries, states, use shapefiles etc.). To install all packages in one command replace the previous command with

```
conda create --name melodies-monet -y -c conda-forge python=3.11
melodies-monet wrf-python jupyterlab regionmask
```

Conda is a very useful tool that lets you create sandboxed environments to contain all of your packages while making sure that you don't break anything outside of it. It also installs all of the required dependencies. To activate your new environment containing MELODIES-MONET and all of its dependencies, run

```
conda activate melodies-monet
```

You should now see, before the prompt in your terminal, `(melodies-monet)`, with the parenthesis.

If your Mac has a Apple Silicon CPUs, these are the instructions (it assumes python is already installed):

```
$ conda create -n melodies-monet -c conda-forge -y python=3.11
melodies-monet jupyterlab
$ conda activate melodies-monet
$ mkdir melodies-monet
$ cd melodies-monet
$ git clone https://github.com/NCAR/wrf-python
$ cd wrf-python
$ pip install .
```

For CAMx, there is a minor bug in MONETIO, the dependency that reads in the model and observational data, when multiple days are read. A fix for it has already been submitted to the main repository, and information on how to apply it manually is provided in the next Section.

[1] wrf-python's conda package is not available for Apple Silicon machines, which are officially unsupported for the package. However, we have been successful compiling it from source. Check https://wrf-python.readthedocs.io/en/latest/installation.html for instructions.

Installing local/development versions

For U.S. State Implementation Plan (SIP) applications, you might want to use a different version (branch) of MELODIES MONET than the default repository, specifically our NSF NCAR ACOM SIP specific version (https://github.com/NCAR/MELODIES-MONET-ACOM/tree/SIP_Colorado). To install it, after following the previous steps, you will need to do some extra work. Our version of the code is in a repository in GitHub. To use it, you will need to use `git`.
Make sure that your `conda` environment is activated, and activate it if it is not.
It is very likely that you already have `git` installed in your system. If you don't, you can install it with conda, by issuing the command

```
conda install -c conda-forge git
```

You do not really need to understand git to follow the next instructions. However, if you'd like to, GitHub (an online repository built using git that hosts the MELODIES-MONET code) offers nice tutorials. Feel free to check out https://docs.github.com/en/get-started.

If you are using the default repository, which is publicly available, there is no need to set up a GitHub account However, our NSF NCAR ACOM repository, which is forked from the official MELODIES-MONET code but adds some specific tools for use with the Colorado State Implementation Plan project and also includes most recent updates not yet merged into the main repository, is currently private. **If you intend to use it, you will need to create a GitHub account and request access by sending an email with your github username to plichtig@ucar.edu or pfister@ucar.edu.** We expect to be able to make it publicly available at some point but most of the code will always also be kept upstream, in the main repository.
The authentication in GitHub can be a bit cumbersome, especially the first time you do it. There are a few ways to do it: you can either use tokens (which are basically a password on steroids) or an `ssh` key. This last method might seem a little bit mysterious and is harder the first time, but it has an advantage: you only need to do it **once per computer**. To learn how to set an ssh key, check out https://docs.github.com/en/authentication/connecting-to-github-with-ssh .

Once you have git installed and access to our ACOM repository, create a directory, e.g called MELODIES_MONET_SIP and from within that directory run:

```
git clone git@github.com:NCAR/MELODIES-MONET-ACOM MELODIES-MONET
cd MELODIES-MONET
git checkout SIP_Colorado
pip install --force-reinstall --no-deps --editable .
```

This will create a subdirectory called MELODIES-MONET containing the code, and will also install the code. The `--editable` flag will make sure that any modifications to the code that you might make are read correctly.

The same methodology can be applied to any library (e.g. monet and monetio) and it is

recommended to do this with the MONETIO dependency. Issue from within your main directory:

```
git clone git@github.com:noaa-oar-arl/monetio.git monetio
cd monetio
git checkout develop
pip install --force-reinstall --no-deps --editable .
```

Bug fix for the CAMx reader

If the bugfix has not been merged into the main code yet, and you need to read multiple CAMx output files, make sure to do the previous steps to install the develop branch. Then do

```
cd monetio/monetio/models
```

Open the file _camx_mm.py with your favorite text editor, and add the following lines in line 633 and 634

```
    keywords["combine"] = "nested"
    keywords["concat_dim"] = "TSTEP"
```

Make sure to maintain the same indentation of the else statement in line 627: i.e., four spaces.

Using Python

As we have mentioned, MELODIES-MONET is built as Python code. Although **you do not require Python knowledge to use it**, knowing enough to be able to read and do minor changes to the code can be useful. If you are unfamiliar with it but are interested in learning, we recommend reading https://foundations.projectpythia.org/landing-page.html .

If you have to choose one and only one Python package to learn the basics, let it be `matplotlib`. Knowing the basics will be helpful to customize your plots. Check out the `matplotlib` section of Project Pythia.

Jupyter-lab

Jupyter-lab is a very practical way to interact with Python code. You can, and sometimes probably should, write a straight up Python script. However, especially while testing or refining, Jupyter will seriously make your life easier.

If running on an HPC system, it is likely that you have access to a JupyterHub, which has already jupyter available. Check your system's documentation or ask your system administrator about it.

In that case, run

```
conda install -c conda-forge ipykernel
```

to gain access to your conda environment.

If you are running on your local machine, make sure that you activate your conda environment (`conda activate melodies-monet`) and run, from the command line, `jupyter-lab`. This will open a tab in your default browser, pointing to a local file. It is not browsing the internet.

Jupyter-lab lets you write blocks of code that can be run one by one (called “cells”), and provides you with graphical updates. You can intercalate them with markdown cells, which provide nicely formatted text, and are not run as code. If you search online, you might find a lot of information about “jupyter notebooks”, which are pretty much the same thing, albeit with a somewhat older interface. The jupyter files end with the extension `.ipynb` (Interactive Python Notebook).
You will find a lot of examples of jupyter notebooks inhttps://github.com/NCAR/MELODIES-MONET/tree/main/docs/examples, and a simple tutorial in https://www.dataquest.io/blog/jupyter-notebook-tutorial/.
Looking at an example is probably going to suffice.

Using YAML files

MELODIES-MONET input is managed using a YAML (YAML Ain't Markup Language™) file. A YAML file is a normal text file, formatted in a specific way.
The input is organized in blocks, determined by indentation. Repeat three times: **indentation matters, indentation matters, indentation matters**. Most of the user problems we have found in MELODIES-MONET are the consequences of wrong indentation.
Comments written in a line after `#` symbols are ignored. There are multiple examples in https://github.com/NCAR/MELODIES-MONET/tree/main/docs/examples
An example (simplified) from the MELODIES-MONET docs.

```
analysis:
  start_time: "2019-09-09 00:00"
  end_time: "2019-09-10 00:00"
  output_dir: ./output/idealized
  save:
    paired:
      method: 'netcdf' # 'netcdf' or 'pkl'
      prefix: 'asdf'
      data: 'all'
  read:
    paired:
      method: 'netcdf' # 'netcdf' or 'pkl'
      filenames:
      test_obs_test_model: 'asdf_test_obs_test_model.nc4'

model:
  test_model:
    files: test_model.nc
    mod_type: random
    variables:
      A:
        units: "Units of A"
```

```
        unit_scale: 1
        unit_scale_method: "*"

obs:
  test_obs:
    # use_airnow: True
    filename: test_obs.nc
    obs_type: pt_sfc

plots:
  plot_grp1:
      type: 'spatial_overlay'
      fig_kwargs:
        states: True
        figsize: [10, 5]
      domain_type: ['all']  # required
      domain_name: ['CONUS']  # required
      data: ['test_obs_test_model']  # required
      data_proc:
        rem_obs_nan: True
        set_axis: True
```

Here you have 4 large blocks, called `analysis`, `models`, `obs` and `plots`. Within the block `analysis`, this file is setting `start_time`, `end_time` and `output_dir`, and 2 sublocks: `save` and `pair`.

Structure of a typical MELODIES-MONET workflow

A typical MELODIES-MONET script consists of the following 9 lines of Python code:

```
1. from melodies_monet import driver
2. an = driver.analysis()
3. an.control = path_to_yaml_file.yaml
4. an.read_control()
5. an.open_obs()
6. an.open_models()
7. an.pair_data()
8. an.plotting()
9. an.stats()
```

Let’s analyze line by line.

1. Loads the “driver” into the code. Driver is a big set of Python code containing everything we need.
2. Creates an instance of “analysis”, which will let us do every operation, and names it “an”.
3. Tells “an” where the control YAML file is set up.
4. Makes “an” read the control YAML file.
5. “an” opens the observations that were set in the YAML file, and does any necessary processing.
6. “an” opens the models that were set in the YAML file, and does any necessary processing.
7. “an” pairs the data, and creates a “pair” object that we can access as an.paired.
8. Does all the necessary plots, as set in the YAML file.
9. Calculates all the statistics, as set in the YAML file.

Additional Resources:

- Example YAML files for SIP evaluation are available in the MELODIES-MONET-ACOM fork (https://github.com/NCAR/MELODIES-MONET-ACOM/tree/SIP_Colorado/sip_specific_tools/examples)
- MELODIES MONET Tutorial at NSF NCAR in October 2024; slides and recordings are available at: https://www2.acom.ucar.edu/events/melodies-tutorial-2024
- A package including example input data, yaml files and run scripts can be shared on request.