MAUD Documentation

Release 0.10.0

Guilherme Castelão & Bia Villas Boas

May 06, 2017

Contents

1 Welcome to MAUD's documentation! 1

1.1 MAUD . . . 1
1.2 Installation . . . 1
1.3 Getting Started with MAUD . . . 2
1.4 Contributing . . . 4
1.5 History . . . 6
1.6 0.10 (2016, Jul) . . . 6
1.7 0.6.1 . . . 6
1.8 0.5.2 . . . 6
1.9 0.4.1 . . . 6
1.10 2003 . . . 6

2 Indices and tables 7

CHAPTER 1

Welcome to MAUD's documentation!

MAUD stands for Moving Averages for Uneven Data. The major characteristic of this package is to expect an uneven distribution of the data. In the case of regular geographic grid, i.e. regular latitudes and longitudes, the weight windows will be defined based on the distance.

Contents:

MAUD

Moving Average for Uneven Data

Maud was also a ship built by Amundsen. http://en.wikipedia.org/wiki/Maud_(ship). Dr. Sverdrup was first a meteorologist, but changed interest to oceanography while in charge of the scientific work on the north pole expedition leaded by Amundsen, aboard the Maud (Cushman-Roisin, 1994).

- Free software: BSD license
- Documentation: https://maud.readthedocs.io.

Features

- TODO

Credits

Installation

Requirements

- Python 2.6 (>=2.6.5), 2.7, 3.1 or 3.2

- Numpy

Strongly recommended:

- distribute 0.6 (>=0.6.40)
- Cython (>= 0.20)

Stable release

Optional: MAUD can run without Cython, but it's strongly recommended since some filters can run more than 10 times faster with Cython. In that case it must be installed before install MAUD itself. To do so, run this command in your terminal:

```
$ pip install distribute>=0.6.40
$ pip install cython
```

To install MAUD, run this command in your terminal:

```
$ pip install maud
```

This is the preferred method to install MAUD, as it will always install the most recent stable release. Cython is not required, but it's strongly recommend since some of MAUD filters can run more than 10 times faster.

If you don't have pip installed, this Python installation guide can guide you through the process.

From sources

The sources for MAUD can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/castelao/maud
```

Or download the tarball:

```
$ curl  -OL https://github.com/castelao/maud/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

Getting Started with MAUD

One can use MAUD inside Python or in the shell.

Shell script

MAUD provides two shell commands, maud4nc for 1D filter and maud4latlonnc for 2D filter on geographic coordinates.

maud4nc

To check the available options:

```
>>> maud4nc -h
```

In the example below, the variable temperature (–var), at the netCDF file model_output.nc, is filtered along the time (–scalevar) using a hann window (-w), and the output will be saved at model_highpass.nc (-o). This is a bandpass filter (–highpasswindowlength together with –lowpasswindowlength), preserving scales between 120 and 10 units of the scalevar (on this case: time).:

```
>>> maud4nc --highpasswindowlength=120 --lowpasswindowlength=10 --scalevar=time \
>>> --var='temperature' -w hann -o model_highpass.nc model_output.nc
```

maud4latlonnc

To check the available options:

```
>>> maud4latlonnc -h
```

In the example below, the variable temperature (–var), at the netCDF file model_output.nc, is filtered along the space (lat x lon). The variables latitude and longitude must exist in the same file. This is a lowpass filter (–largerpasslength), hence it attenuates eveything with spatial scale smaller than 600e3 meters. The weights are defined by a hamming function (-w). The npes define the number of parallel process to be used, in this case 18. The option –interp defines that any missing value will be replaced in the output as the filtered result of the valid values around it, inside the window lenght.:

```
>>> maud4latlonnc --largerpasslength=600e3 --var='temperature' \
>>> -w hamming --interp --npes=18 -o model_highpass model_output.nc
```

Inside Python

```
>>> from maud import window_1Dmean, window_mean_2D_latlon
>>> window_1Dmean(x, l=200e3, t=None, method='hann', axis=0, parallel=True)
```

```
>>> window_mean_2D_latlon(Lat, Lon, data, l, method='hamming', interp=False)
```

The faster version

There is a Cython version of each filter. If you're able to, use cmaud instead of maud to gain at least one order of magnitude on the speed.

```
>>> from cmaud import window_1Dmean
>>> window_1Dmean(x, l=200e3)
```

```
>>> from cmaud import window_mean_2D_latlon
>>> window_mean_2D_latlon(Lat, Lon, data, l)
```

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/castelao/maud/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

MAUD could always use more documentation, whether as part of the official MAUD docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/castelao/maud/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *maud* for local development.

1. Fork the *maud* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/maud.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv maud
$ cd maud/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 maud tests
$ py.test tests
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/castelao/maud/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test tests/test_hamming.py
```

History

0.10 (2016, Jul)

- Cleanning up and restructuring MAUD.

0.6.1

- maud4nc, a shell script to filter data on the NetCDF itself.

0.5.2

- Including some resources with cython, hence faster.
- You must have the package distribute with a newer version than 0.6.24dev-r0, otherwise it will fail to install

0.4.1

- window_1Dmean is now recursive and uses multiprocessing if it has more than 1 dimension.
- Sorry, lost the track before this.

2003

- MAUD was resulted from a set of functions from my Master's Degree, when I needed to handle PIRATA time series, which is characterized by regular data gaps.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search