ESMLab Documentation

Release 2019.4.27

Earth System Informatics Team

Apr 27, 2019

CONTENTS:

Tools for working with earth system multi-model analyses with xarray. See documentation for more information.

CHAPTER

ONE

INSTALLATION

ESMLab can be installed from PyPI with pip:

```
pip install esmlab
```

It is also available from *conda-forge* for conda installations:

```
conda install -c conda-forge esmlab
```

1.1 Installing

You can install esmlab with `pip`, `conda`, or by installing from source.

1.1.1 Pip

Pip can be used to install esmlab:

```
pip install esmlab
```

1.1.2 Conda

To install the latest version of esmlab from the conda-forge repository using conda:

```
conda install -c conda-forge esmlab
```

1.1.3 Install from Source

To install esmlab from source, clone the repository from github:

```
git clone https://github.com/NCAR/esmlab.git
cd esmlab
pip install -e .
```

You can also install directly from git master branch:

```
pip install git+https://github.com/NCAR/esmlab
```

1.1.4 Test

To run esmlab's tests with `pytest`:

```
git clone https://github.com/NCAR/esmlab.git
cd esmlab
pytest -v
```

1.2 API Reference

This page provides an auto-generated summary of esmlab's API. For more details and examples, refer to the relevant chapters in the main part of the documentation.

1.2.1 Utilities

datasets.open_dataset(name[, cache, ...])	Load a dataset from the online repository (requires access to internet).

1.2.2 Configuration

config.get(key[, default, config])	Get elements from global config
config.set([arg, config, lock])	Temporarily set configuration values within a context manager

1.2.3 Statistics functions

statistics.weighted_sum(data[, dim, weights])	Compute weighted sum for xarray objects
statistics.weighted_mean(data[, dim, weights])	Compute weighted mean for xarray objects
statistics.weighted_std(data[, dim, weights])	Compute weighted standard deviation for xarray objects
statistics.weighted_rmsd(x, y[, dim, weights])	Compute weighted root mean square deviation between two xarray DataArrays
statistics.weighted_cov(x, y[, dim, weights])	Compute weighted covariance between two xarray DataArrays.
statistics.weighted_corr(x, y[, dim, ...])	Compute weighted correlation between two *xarray.DataArray*.

1.2.4 Climatologies

core.climatology(dset, freq[, time_coord_name])	Compute climatologies for a specified time frequency

1.2.5 Anomalies

core.anomaly(dset, clim_freq[, ...])	Compute anomalies for a specified time frequency

1.2.6 Resample

core.resample(dset, freq[, weights, ...])	Resample given dataset and computes the mean for specified sampling time frequecy

esmlab.datasets.**open_dataset**(*name*, *cache=True*, *cache_dir='/home/docs/.esmlab/sample_data'*, *github_url='https://github.com/NCAR/esmlab-data'*, *branch='master'*, ***kwargs*)

Load a dataset from the online repository (requires access to internet).

If a local copy is found then always use that to avoid network traffic.

Parameters

name [str] Name of the netcdf file containing the dataset ie. 'air_temperature'

cache_dir [string, optional] The directory in which to search for and write cached data.

cache [boolean, optional] If True, then cache data locally for use on subsequent calls

github_url [string] Github repository where the data is stored

branch [string] The git branch to download from

kwargs [dict, optional] Passed to xarray.open_dataset

esmlab.config.**get**(*key*, *default='__no_default__'*, *config={'esmlab': {'sample-data-dir': '/home/docs/.esmlab/sample_data'}}*)

Get elements from global config

Use '.' for nested access

See also:

esmlab.config.set

Examples

```
>>> from esmlab import config
>>> config.get('foo')  # doctest: +SKIP
{'x': 1, 'y': 2}
```

```
>>> config.get('foo.x')  # doctest: +SKIP
1
```

```
>>> config.get('foo.x.y', default=123)  # doctest: +SKIP
123
```

esmlab.config.**set**(*arg=None*, *config={'esmlab': {'sample-data-dir': '/home/docs/.esmlab/sample_data'}}*, *lock=<unlocked _thread.lock object>*, ***kwargs*)

Temporarily set configuration values within a context manager

See also:

esmlab.config.get

Examples

```
>>> import esmlab
>>> with esmlab.config.set({'foo': 123}):
...     pass
```

esmlab.config.**collect**(*paths=['/home/docs/.esmlab'], env=None*)
Collect configuration from paths and environment variables

Parameters

paths [List[str]] A list of paths to search for yaml config files

env [dict] The system environment variables

Returns

config: dict

See also:

esmlab.config.refresh collect configuration and update into primary config

esmlab.config.**refresh**(*config={'esmlab': {'sample-data-dir': '/home/docs/.esmlab/sample_data'}}, defaults=[], **kwargs*)
Update configuration by re-reading yaml files and env variables

This mutates the global esmlab.config.config, or the config parameter if passed in.

This goes through the following stages:

1. Clearing out all old configuration
2. Updating from the stored defaults from downstream libraries (see update_defaults)
3. Updating from yaml files and environment variables

Note that some functionality only checks configuration once at startup and may not change behavior, even if configuration changes. It is recommended to restart your python process if convenient to ensure that new configuration changes take place.

See also:

esmlab.config.collect for parameters

esmlab.config.update_defaults

esmlab.statistics.**weighted_sum**(*data*, *dim=None*, *weights=None*)
Compute weighted sum for xarray objects

Parameters

data [xarray.Dataset or xarray.DataArray] xarray object for which to compute *weighted sum*

dim [str or sequence of str, optional] Dimension(s) over which to apply sum.

weights [xarray.DataArray or array-like] weights to apply. Shape must be broadcastable to shape of data.

Returns

reduced [xarray.Dataset or xarray.DataArray] New xarray object with weighted sum applied to its data and the indicated dimension(s) removed.

esmlab.statistics.**weighted_mean**(*data*, *dim=None*, *weights=None*)
Compute weighted mean for xarray objects

Parameters

data [xarray.Dataset or xarray.DataArray] xarray object for which to compute *weighted mean*

dim [str or sequence of str, optional] Dimension(s) over which to apply mean.

weights [xarray.DataArray or array-like] weights to apply. Shape must be broadcastable to shape of data.

Returns

reduced [xarray.Dataset or xarray.DataArray] New xarray object with weighted mean applied to its data and the indicated dimension(s) removed.

esmlab.statistics.**weighted_std**(*data*, *dim=None*, *weights=None*)
Compute weighted standard deviation for xarray objects

Parameters

data [xarray.Dataset or xarray.DataArray] xarray object for which to compute *weighted std*

dim [str or sequence of str, optional] Dimension(s) over which to apply standard deviation.

weights [xarray.DataArray or array-like] weights to apply. Shape must be broadcastable to shape of data.

Returns

reduced [xarray.Dataset or xarray.DataArray] New xarray object with weighted standard deviation applied to its data and the indicated dimension(s) removed.

esmlab.statistics.**weighted_rmsd**(*x*, *y*, *dim=None*, *weights=None*)
Compute weighted root mean square deviation between two xarray DataArrays

Parameters

x, y [xarray.DataArray] xarray DataArray for which to compute *weighted_rmsd*.

dim [str or sequence of str, optional] Dimension(s) over which to apply rmsd.

weights [xarray.DataArray or array-like] weights to apply. Shape must be broadcastable to shape of data.

Returns

reduced [xarray.DataArray] New DataArray with root mean square deviation applied to x, y and the indicated dimension(s) removed.

esmlab.statistics.**weighted_cov**(*x*, *y*, *dim=None*, *weights=None*)
Compute weighted covariance between two xarray DataArrays.

Parameters

x, y [xarray.DataArray] xarray DataArrays for which to compute *weighted covariance*.

dim [str or sequence of str, optional] Dimension(s) over which to apply covariance.

weights [xarray.DataArray or array-like] weights to apply. Shape must be broadcastable to shape of data.

Returns

reduced [DataArray] New DataArray with covariance applied to x, y and the indicated dimension(s) removed.

esmlab.statistics.**weighted_corr**(*x*, *y*, *dim=None*, *weights=None*, *return_p=True*)
Compute weighted correlation between two *xarray.DataArray*.

Parameters

x, y [xarray.DataArray] xarray DataArrays for which to compute *weighted correlation*.

dim [str or sequence of str, optional] Dimension(s) over which to apply correlation.

weights [xarray.DataArray or array-like] weights to apply. Shape must be broadcastable to shape of data.

return_p [bool, default: True] If True, compute and return the p-value(s) associated with the correlation.

Returns

reduced [xarray.DataArray] New DataArray with correlation applied to x, y and the indicated dimension(s) removed.

If *return_p* is True, appends the resulting p values to the returned Dataset.

esmlab.core.**climatology**(*dset*, *freq*, *time_coord_name=None*)
Compute climatologies for a specified time frequency

Parameters

dset [xarray.Dataset] The data on which to operate

freq [str] Frequency alias. Accepted alias:

- `mon`: for monthly climatologies

time_coord_name [str] Name for time coordinate to use

Returns

computed_dset [xarray.Dataset] The computed climatology data

esmlab.core.**anomaly**(*dset*, *clim_freq*, *slice_mon_clim_time=None*, *time_coord_name=None*)
Compute anomalies for a specified time frequency

Parameters

dset [xarray.Dataset] The data on which to operate

clim_freq [str] Climatology frequency alias. Accepted alias:

- `mon`: for monthly climatologies

slice_mon_clim_time [slice, optional] a slice object passed to *dset.isel(time=slice_mon_clim_time)* for subseting the time-period overwhich the climatology is computed

time_coord_name [str] Name for time coordinate to use

Returns

computed_dset [xarray.Dataset] The computed anomaly data

esmlab.core.**resample**(*dset*, *freq*, *weights=None*, *time_coord_name=None*)
Resample given dataset and computes the mean for specified sampling time frequecy

Parameters

dset [xarray.Dataset] The data on which to operate

freq [str] Time frequency alias. Accepted aliases:

- `mon`: for monthly means
- `ann`: for annual means

weights [array_like, optional] weights to use for each time period. This argument is supported for annual means only! If None and dataset doesn't have *time_bound* variable, every time period has equal weight of 1.

time_coord_name [str] Name for time coordinate to use

Returns

computed_dset [xarray.Dataset] The resampled data with computed mean

1.3 Contribution Guide

Interested in helping build Esmlab? Have code from your research that you believe others will find useful? Have a few minutes to tackle an issue?

Contributions are highly welcomed and appreciated. Every little help counts, so do not hesitate!

The following sections cover some general guidelines regarding development in esmlab for maintainers and contributors. Nothing here is set in stone and can't be changed. Feel free to suggest improvements or changes in the workflow.

Contribution links

1.3.1 Feature requests and feedback

We'd also like to hear about your propositions and suggestions. Feel free to submit them as issues and:

- Explain in detail how they should work.
- Keep the scope as narrow as possible. This will make it easier to implement.

1.3.2 Report bugs

Report bugs for esmlab in the issue tracker.

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, specifically the Python interpreter version, installed libraries, and esmlab version.
- Detailed steps to reproduce the bug.

If you can write a demonstration test that currently fails but should pass (xfail), that is a very useful commit to make as well, even if you cannot fix the bug itself.

1.3.3 Fix bugs

Look through the GitHub issues for bugs.

Talk to developers to find out how you can fix specific bugs.

1.3.4 Write documentation

Esmlab could always use more documentation. What exactly is needed?

- More complementary documentation. Have you perhaps found something unclear?
- Docstrings. There can never be too many of them.
- Blog posts, articles and such – they're all very appreciated.

You can also edit documentation files directly in the GitHub web interface, without using a local copy. This can be convenient for small fixes.

Note:

Build the documentation locally with the following command:

```
$ conda env update -f ci/environment-dev-3.7.yml
$ cd docs
$ make html
```

The built documentation should be available in the `docs/_build/`.

1.3.5 Preparing Pull Requests

1. Fork the esmlab GitHub repository. It's fine to use `esmlab` as your fork repository name because it will live under your user.
2. Clone your fork locally using git, connect your repository to the upstream (main project), and create a branch:

```
$ git clone git@github.com:YOUR_GITHUB_USERNAME/esmlab.git
$ cd esmlab
$ git remote add upstream git@github.com:NCAR/esmlab.git

# now, to fix a bug or add feature create your own branch off "master":

$ git checkout -b your-bugfix-feature-branch-name master
```

If you need some help with Git, follow this quick start guide: https://git.wiki.kernel.org/index.php/QuickStart

3. Install dependencies into a new conda environment:

```
$ conda env update -f ci/environment-dev-3.7.yml
$ conda activate esmlab-dev
```

4. Make an editable install of esmlab by running:

```
$ pip install -e .
```

5. Install pre-commit and its hook on the esmlab repo:

```
$ pip install --user pre-commit
$ pre-commit install
```

Afterwards `pre-commit` will run whenever you commit.

https://pre-commit.com/ is a framework for managing and maintaining multi-language pre-commit hooks to ensure code-style and code formatting is consistent.

> Now you have an environment called `esmlab-dev` that you can work in. You'll need to make sure to activate that environment next time you want to use it after closing the terminal or your system.

6. Run all the tests

Now running tests is as simple as issuing this command:

```
$ pytest --junitxml=test-reports/junit.xml --cov=./
```

This command will run tests via the "pytest" tool against Python 3.7.

7. Create a new changelog entry in `CHANGELOG.rst`:

- The entry should be entered as:

  <description> (:pr:`#<pull request number>`)`<author's names>`_

  where `<description>` is the description of the PR related to the change and `<pull request number>` is the pull request number and `<author's names>` are your first and last names.

- Add yourself to list of authors at the end of `CHANGELOG.rst` file if not there yet, in alphabetical order.

8. You can now edit your local working copy and run the tests again as necessary. Please follow PEP-8 for naming.

When committing, `pre-commit` will re-format the files if necessary.

9. Commit and push once your tests pass and you are happy with your change(s):

```
$ git commit -a -m "<commit message>"
$ git push -u
```

10. Finally, submit a pull request through the GitHub website using this data:

```
head-fork: YOUR_GITHUB_USERNAME/esmlab
compare: your-branch-name

base-fork: NCAR/esmlab
base: master          # if it's a bugfix or feature
```

1.4 Release Procedure

Our current policy for releasing is to aim for a release every two weeks. The idea is to get fixes and new features out instead of trying to cram a ton of features into a release and by consequence taking a lot of time to make a new one.

1. Create a new branch `release-yyyy.mm.dd` with the version for the release

- Update *CHANGELOG.rst*
- Make sure all new changes, features are reflected in the documentation.

1. Open a new pull request for this branch targeting *master*
2. After all tests pass and the PR has been approved, merge the PR into `master`
3. Tag a release and push to github:

```
$ git tag -a v2019.1.1 -m "Version 2019.1.1"
$ git push origin master --tags
```

4. Build and publish release on PyPI:

```
$ git clean -xfd  # remove any files not checked into git
$ python setup.py sdist bdist_wheel --universal  # build package
$ twine upload dist/*  # register and push to pypi
```

5. Update esmlab conda-forge feedstock
    - Fork esmlab-feedstock repository
    - Clone this fork and edit recipe:

```
$ git clone git@github.com:username/esmlab-feedstock.git
$ cd esmlab-feedstock
$ cd recipe
$ # edit meta.yaml
```

    - Update version
    - Get sha256 from pypi.org for esmlab
    - Fill in the rest of information as described here
    - Commit and submit a PR

1.5 Changelog History

1.5.1 Esmlab v2019.4.27 (2019-04-27)

Features

- Enable computing significance metrics (p-value) for `weighted_corr`. (GH#90) Riley Brady
- Drop support for Python 2. (GH#109) Anderson Banihirwe
- Nomenclature change: `compute_mon_climatology`, `compute_mon_anomaly`, `compute_ann_mean` functions are now renamed to `climatology`, `anomaly`, and `resample`. (GH#109) Anderson Banihirwe
- Add `.esmlab` accessor allowing users to access functions such as climatology with `dset.esmlab.set_time().climatology()` syntax. (GH#109) Anderson Banihirwe
- Add `compute_mon_mean` function to compute monthly averages. (GH#110) Alper Altuntas

Bug Fixes

- Fix bug that made computing annual mean with `resample(ds, freq='ann')` yield incorrect results. (GH#112) (GH#115) Anderson Banihirwe
- Fix time and time_bounds decoding incosistencies in `resample(ds, freq='ann')` results for `decode_time=True`. (GH#111) (GH#115) Anderson Banihirwe
- Fix cftime datetime decoding/encoding (GH#118) (GH#122) Anderson Banihirwe

Trivial/Internal Changes

- Move regridding utilities to esmlab-regrid repo (GH#107) Anderson Banihirwe
- Switch from versioneer to setuptools-scm (GH#120) Anderson Banihirwe

1.5.2 Esmlab v2019.3.16 (2019-03-16)

Features

- Add statistics functions for DataArray and Dataset to `statistics.py` module (GH#97) Anderson Banihirwe

Functions added:

- `weighted_mean`
- `weighted_sum`
- `weighted_std`
- `weighted_covariance`
- `weighted_correlation`

Bug Fixes

- Increase rtol for float32 weights in `statistics.py` module (GH#81) Michael Levy
- Remove duplicate call to `statistics._get_weights_and_dims` (GH#88) Sudharsana K J L
- Fix bugs in `utils.time.time_manager` add tests for climatology corner cases (GH#100) Matthew Long
  - Allow `climatology.compute_ann_mean` to work if time is encoded
  - Make sure `time:calendar` is preserved in `climatolog.compute_ann_mean`

1.5.3 Esmlab v2019.2.28 (2019-02-28)

Features

- Add function to flexibily compute weights and dimensions to use in statistical operations (GH#74) Anderson Banihirwe
- Add `time_manager` class to support managing the time coordinate of datasets (GH#75) and (GH#76) Matthew Long

Bug Fixes

- Remove hard-coded `tb_name=time_bound` in `compute_time_var` (GH#72) Anderson Banihirwe

Documentation

- Add release procedure to documentation (GH#78) Anderson Banihirwe

Trivial/Internal Changes

- Use esmlab-data in tests (GH#67) Anderson Banihirwe
- Update continuous integration workflow (GH#68) Anderson Banihirwe

1.5.4 Esmlab v2019.2.1 (2019-02-12)

- Add `time_bound` to output of `compute_ann_mean` (GH#51) Matthew Long
- Add xarray alignment option to prevent using mismatching coordinates (GH#54) Anderson Banihirwe
- Add regridding functionality (GH#56) Matthew Long
- Handle `time_bound` on data read with `decode_times=True` (GH#59) Matthew Long
- Add interface to esmlab-data (GH#61) Anderson Banihirwe

1.5.5 Esmlab v2019.2.0 (2019-02-02)

- Rename `compute_ann_climatology` to `compute_ann_mean` (GH#33) Anderson Banihirwe
- Don't add `NaNs` for `_FillValue` (GH#34) Anderson Banihirwe
- Change time handling for `compute_mon_climatology` and `compute_ann_mean` (GH#37) Matthew Long
- Add slice_mon_clim_time argument to `compute_mon_climatology` (GH#37) Matthew Long
- Drop `time_bound` variable from `compute_ann_mean` (GH#43) Matthew Long

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX