esmlab-regrid Documentation

Release 2019.5.2.dev12+g5a65521.d20210430

Earth System Informatics Team

Apr 30, 2021

CONTENTS:

Esmlab.regrid is a lightweight library for regridding in Python. It relies on both *xesmf* and *ESMF* functionaly.

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

CHAPTER

ONE

PIP

Pip can be used to install esmlab-regrid:

```
pip install esmlab-regrid
```

CHAPTER

TWO

CONDA

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

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

CHAPTER

THREE

INSTALL FROM SOURCE

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

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

You can also install directly from git master branch:

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

CHAPTER

FOUR

TEST

Test esmlab-regrid with `pytest`:

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

CHAPTER

FIVE

TUTORIAL

`esmlab-regrid` supports a regridding workflow that is based on named grid files saved in SCRIP format.

[1]:
```
%matplotlib inline
import xarray as xr
import numpy as np

import esmlab_regrid
import esmlab
```

5.1 Load some dummy data.

[2]:
```
dsx1 = xr.open_dataset('/glade/work/mclong/grids/POP_gx1v7.nc')
dsx1 = dsx1.drop([v for v in dsx1.variables if v not in ['z_t','HT','KMT']])

dsx3 = xr.open_dataset('/glade/work/mclong/grids/pop-grid-g37.nc')
dsx3 = dsx3.drop([v for v in dsx3.variables if v not in ['z_t','HT','KMT']])

dsx1.HT.values = np.where(dsx1.KMT > 0,dsx1.HT.values,np.nan)
dsx3.HT.values = np.where(dsx3.KMT > 0,dsx3.HT.values,np.nan)
```

[3]:
```
esmlab.config.get('regrid.gridfile-directory')
```

[3]:
```
'/glade/u/home/abanihi/.esmlab/esmlab-grid-files'
```

[4]:
```
esmlab.config.set({'regrid.gridfile-directory': '/glade/work/abanihi/esmlab-regrid'})
```

[4]:
```
<esmlab.config.set at 0x2aaaafa85eb8>
```

[5]:
```
esmlab.config.get('regrid.gridfile-directory')
```

[5]:
```
'/glade/work/abanihi/esmlab-regrid'
```

5.2 Instantiate the `regridding` object

The regridder is initialized with a specific source grid and destination grid. The grid files describeing this must be present (**future work will enable on-the-fly generation**).

[6]:
```
method = 'bilinear'
src_grid_name = 'POP_gx1v7'
dst_grid_name = 'POP_gx3v7'
```

[7]:
```
%%time
R = esmlab_regrid.regridder(name_grid_src=src_grid_name, name_grid_dst=dst_grid_name,
                            method=method, overwrite_existing=False)
```

```
/glade/work/abanihi/esmlab-regrid
/glade/work/abanihi/esmlab-regrid
Generating /glade/work/abanihi/esmlab-regrid/weights/POP_gx1v7_to_POP_gx3v7_bilinear
CPU times: user 2.22 s, sys: 88 ms, total: 2.3 s
Wall time: 2.37 s
```

5.3 Perform the regridding

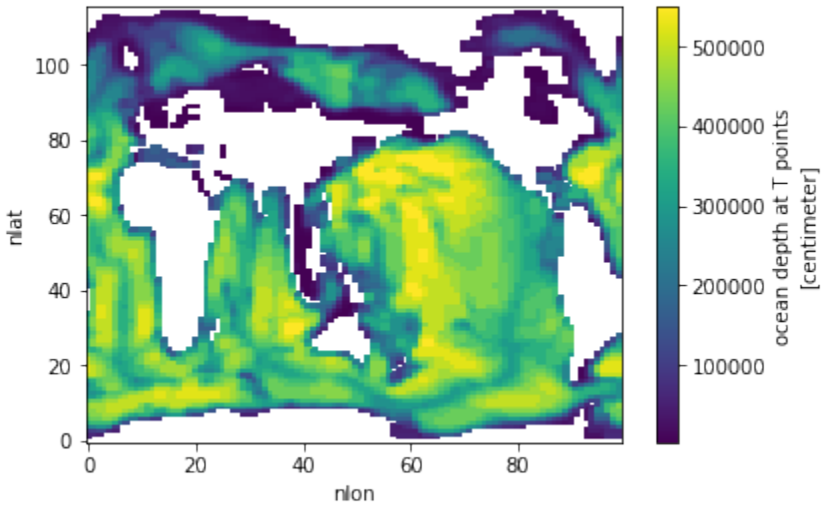
[8]:
```
da_x1_on_x3 = R(dsx1.HT, renormalize=True)
da_x1_on_x3
```

```
/glade/work/abanihi/softwares/miniconda3/envs/analysis/lib/python3.7/site-packages/
↪esmlab_regrid/core.py:203: RuntimeWarning: invalid value encountered in greater
  data_dst = np.where(ones_dst > 0.0, data_dst, np.nan)
```

[8]:
```
<xarray.DataArray 'HT' (nlat: 116, nlon: 100)>
array([[         nan,          nan,          nan, ...,          nan,
                 nan,          nan],
       [82960.6956  ,          nan,          nan, ...,          nan,
        36713.721329, 74816.301981],
       [80975.451175, 69774.160175, 37670.737975, ..., 30221.084719,
        33983.806102, 52968.637226],
       ...,
       [         nan,          nan,          nan, ...,          nan,
                 nan,          nan],
       [         nan,          nan,          nan, ...,          nan,
                 nan,          nan],
       [         nan,          nan,          nan, ...,          nan,
                 nan,          nan]])
Dimensions without coordinates: nlat, nlon
Attributes:
    long_name:      ocean depth at T points
    units:          centimeter
    regrid_method:  bilinear
    history:        \n2019-05-01 13:54:11.162852 esmlab.regrid <regrid>
```

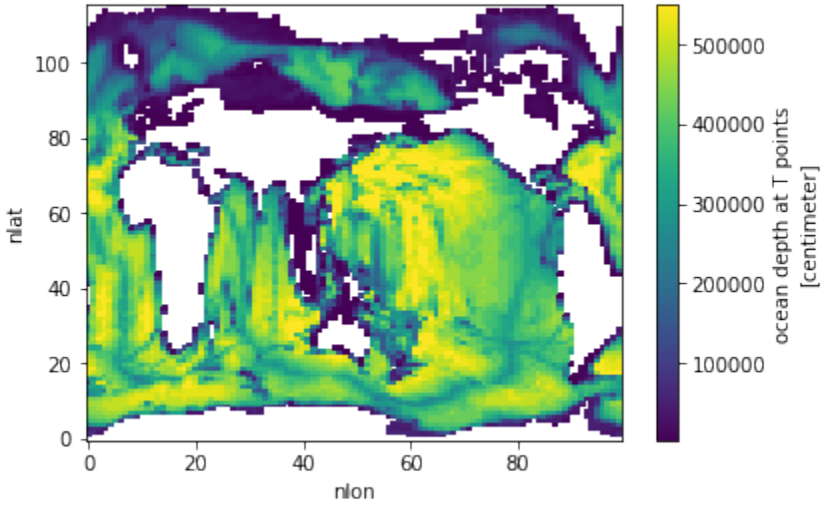
[9]:
```
dsx3.HT.plot()
```

[9]:
```
<matplotlib.collections.QuadMesh at 0x2aab67c7c710>
```



[10]:
```
da_x1_on_x3.plot()
```

[10]:
```
<matplotlib.collections.QuadMesh at 0x2aab6a5b8240>
```



[11]:
```
%load_ext watermark
%watermark --iversion -g  -m -v -u -d
```

```
esmlab         v2019.3.16+140.gd3ce9a5
numpy          1.15.4
esmlab_regrid 0+untagged.19.gb8b7182
xarray         0.12.1
last updated: 2019-05-01

CPython 3.7.3
IPython 7.1.1

compiler   : GCC 7.3.0
system     : Linux
release    : 3.12.62-60.64.8-default
machine    : x86_64
```

(continues on next page)

(continued from previous page)

```
processor  : x86_64
CPU cores  : 72
interpreter: 64bit
Git hash   : 5a8cedcec68d55c4b9dbe48f778b44b444db5297
```

CHAPTER

SIX

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.

Regridder(name_grid_src, name_grid_dst[, ...])	Class to enable regridding between named grids.

class esmlab_regrid.core.**Regridder**(*name_grid_src*, *name_grid_dst*, *method='bilinear'*, *overwrite_existing=False*)

Class to enable regridding between named grids.

__init__(*name_grid_src*, *name_grid_dst*, *method='bilinear'*, *overwrite_existing=False*)

Parameters

name_grid_src [string] Name of source grid.

name_grid_dst [string] Name of destination grid.

method [string, optional] Regridding method. Options are:

- 'bilinear'
- 'conservative'
- 'patch'
- 'nearest_s2d'
- 'nearest_d2s'

overwrite_existing [bool, optional [Default=False]] Overwrite previously generated weight files.

__call__(*data_in*, *renormalize=True*, *apply_mask=True*)

Perform regridding on an *xarray.DataArray* or *xarray.Dataset*.

Parameters

data_in [*xr.DataArray* or *xr.Dataset*] The data to regrid

renormalize [bool, optional [default=True]] Logical flag to trigger renormalization of the remapping weights. This is useful if the remapping weight-file was computed with a different missing value mask than *da_in*. For instance, in regridding 3D ocean data, it is possible to use a mapping file computed at the surface at each successive depth level: setting *renormalize=True* will ensure correct handling of missing values.

apply_mask [bool, optional [default=False]] Apply a missing-values mask after regridding operations.

Returns

data_out [*xr.DataArray* or *xr.Dataset*] The dataarray or dataset regridded to the destination grid.

CHAPTER

SEVEN

CONTRIBUTION GUIDE

Interested in helping build esmlab-regrid? Have code from your work 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-regrid 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

7.1 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.

7.2 Report bugs

Report bugs for esmlab-regrid 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-regrid 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.

7.3 Fix bugs

Look through the GitHub issues for bugs.

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

7.4 Write documentation

esmlab-regrid 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/`.

7.5 Preparing Pull Requests

1. Fork the esmlab-regrid GitHub repository. It's fine to use `esmlab-regrid` 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-regrid.git
$ cd esmlab-regrid
$ git remote add upstream git@github.com:NCAR/esmlab-regrid.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-regrid-dev
```

4. Make an editable install of esmlab-regrid by running:

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

5. Install pre-commit and its hook on the esmlab-regrid 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-regrid-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-regrid
compare: your-branch-name

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

CHAPTER

EIGHT

CHANGELOG HISTORY

8.1 ESMLab-regrid v2019.5.1 (2019-05-01)

- First Release
- The original code for this package came out of esmlab and was contained in `esmlab.regrid` module.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

R