Hands-on session “developers”

Outline

- Contribute to pysteps
- Git flow
- Best practices
- Test the code
- Document the code

Contribute to pysteps

Welcome! Pysteps is a community-driven initiative for developing and maintaining an easy to use, modular, free and open-source Python framework for short-term ensemble prediction systems.

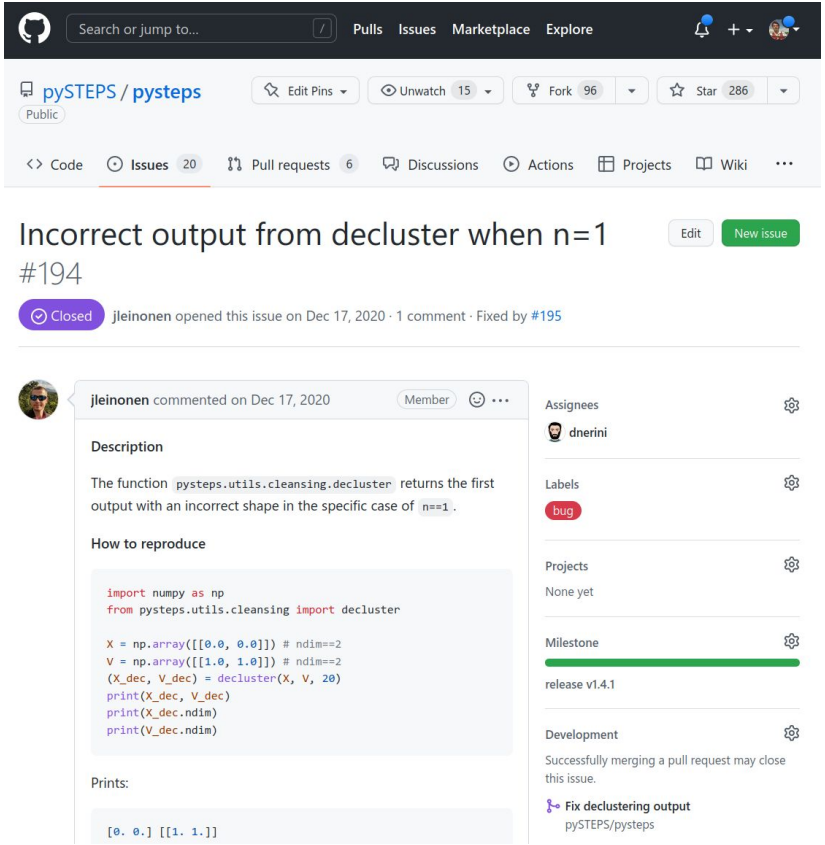
There are many ways to contribute to pysteps:

1. Bug reports and feature requests;
2. Code contributions with new features or bug fixes;
3. Documentation;
4. Writing examples;
5. Writing tests;
6. Providing feedback.

xkcd.com/1822

Step 0: Open an issue

1. Explain what you want to do and why.
2. Discuss how you plan to do it.
3. Make sure to start on the right track (and avoid dead ends).
4. Follow templates, e.g. bug reports:
    a. clear and descriptive title
    b. steps to reproduce
    c. obtained result
    d. expected result



Git flow

1. **Create a branch**

- Create a new branch in your repository.
- A short, descriptive branch name enables your collaborators to see ongoing work at a glance. For example, *new-blending-weights*, *testing-probabilistic-nowcasting*.

Git flow

1. **Create a branch**
2. **Make changes/improvements/developments**

   Your branch is a safe place to make changes.

   If you make a mistake, you can revert your changes or push additional changes to fix the mistake.

   Your changes will not end up on the default branch until you merge your branch.

xkcd.com/1597

Git flow

1. **Create a branch**
2. **Make changes/improvements/developments**
   a. **commits**

- Any file addition, edit, or deletion is a commit.
- Each commit is an individual unit of change – this makes it easier to roll back a given change if necessary.
- Write clear commit messages.
  git commit -m «Concise commit message» -m «A more verbose message»

Git flow

1. **Create a branch**
2. **Make changes/improvements/developments**
3. **Create a pull request**

Create a pull request to ask collaborators for feedback on your changes. Pull request review is so valuable that some repositories require an approving review before pull requests can be merged. If you want early feedback or advice before you complete your changes, you can mark your pull request as a draft.

NOTE: The work is not done yet, here it comes the most important/interactive part!!!

Git flow

1. **Create a branch**
2. **Make changes/improvements/developments**
3. **Create a pull request**
4. **Address review comments**

Reviewers may leave questions, comments, and suggestions.

Reviewers can comment on the whole pull request or specific lines.

The reviewers, as well as yourself, can insert images or code suggestions to clarify the comments.

You can continue to commit and push changes in response to the reviews. *Your pull request will update automatically.*

Examples of review comments/process

https://github.com/pySTEPS/pysteps/pull/229

Examples of review comments

Linear blending #229

Merged | RubenImhoff merged 8 commits into pySTEPS:pysteps-v2 from wdewettin:linear_blending on 3 Nov 2021

Conversation 30 | Commits 8 | Checks 0 | Files changed 8 | +568 −2

wdewettin commented on 9 Aug 2021

This addresses Issue #226. I added a nowcasting method called linear blending. This works for all nowcast methods (deterministic and ensemble) and is also compatible with NWP ensembles. Furthermore, I added an example which uses some dummy NWP data to blend with and which plots the results. Lastly, I also added a test for the linear blending function.

wdewettin and others added 3 commits 13 months ago

- Implemented linear blending ... 88c4d4b
- Replaced missing data in nowcasting with NWP data ... f81f4f2
- Added compatibility for NWP ensembles 7d0d081

RubenImhoff self-requested a review 13 months ago

RubenImhoff added the enhancement label on 9 Aug 2021

RubenImhoff added this to In progress in pysteps-v2 via automation on 9 Aug 2021

RubenImhoff linked an issue on 9 Aug 2021 that may be closed by this pull request

Implement linear blending #226 — Closed

Merge branch 'pysteps-v2' of github.com:wdewettin/pysteps into linear… ... d6d0638

RubenImhoff requested review from dnerini and cvelascof 13 months ago

Reviewers: ladc, cvelascof, RubenImhoff, dnerini

Assignees: No one assigned

Labels: enhancement

Projects: pysteps-v2 — Done

Milestone: No milestone

Development: Successfully merging this pull request may close these issues. — Implement linear blending

5 participants

RubenImhoff commented on 9 Aug 2021 — Contributor

Great work, @wdewettin! I was (actually, we were - that is including @wdewettin and @ladc) wondering whether we should place this method in a/the blending folder or whether it fits in the nowcasts folder as currently done by @wdewettin. Any thoughts?

RubenImhoff reviewed on 9 Aug 2021 — View changes

RubenImhoff left a comment — Contributor

Great, @wdewettin! I left some comments and questions. Some of them I could implement myself as well. Let's just wait for the comments of the others (and feel free to comment yourself on them), then we'll continue with the final changes. Exciting to have this option in pysteps soon!

pysteps/nowcasts/linear_blending.py — Outdated

```
52 +     containing the NWP precipitation fields ordered by timestamp from oldest
53 +     to newest. The time steps between the inputs are assumed to be regular
54 +     (and identical to the time step between the nowcasts). If no NWP
55 +     data is given the value of R_nwp is None and no blending will be performed.
```

RubenImhoff on 9 Aug 2021 — Contributor

In case of R_nwp = None, the script is similar to steps.py, right? Do we want to use the script as a postprocessing tool, i.e. the user makes a nowcast with steps.py and passes the output on to linear_blending.py, or do we want to keep it this way and have some duplicate code with steps.py? As steps.py may also change to allow for other blending methods, I wonder what everyone thinks about the best way forward.

pysteps/nowcasts/linear_blending.py — Outdated

```
82 +     )
83 +
84 +     # Transform the precipitation back to mm/h
85 +     R_nowcast = transformation.dB_transform(R_nowcast, threshold=-10.0, inverse=True)[0]
```

RubenImhoff on 9 Aug 2021 — Contributor

I believe the threshold is present in the metadata (the transformer at least returns the metadata). We could add this to the function, then we don't have to hard code it here.

cvelascof on 1 Oct 2021 — Member

I understand why this is done here, but this is assuming that `R_nwp` is in mm/hr and that the `precip` is in dB. If this is the case, then we need to indicate those limitations in the description of the function. Alternatively, both sources should be in `mm/hr` and the conversion of `precip` to `dB` be done within this function as well.

cvelascof on 1 Oct 2021 — Member

I understand why this is done here, but this is assuming that `R_nwp` is in mm/hr and that the `precip` is in dB. If this is the case, then we need to indicate those limitations in the description of the function. Alternatively, both sources could be in `mm/hr` and the conversion of `precip` to `dB` be done within this function as well.

Said that the returned output `R_blended` will be always in `mm/hr`

pysteps/nowcasts/linear_blending.py — Outdated

```
112 +                 R_nowcast[np.newaxis, :, :], n_ens_members_max, axis=0
113 +             )
114 +         else:
115 +             repeats = [
```

RubenImhoff on 9 Aug 2021 — Contributor

After consulting @wdewettin and @ladc, I figured out what this piece of code does. So if you e.g. have 10 ensemble nowcasts members and 3 NWP members, the output will be an ensemble of 10 members. Hence, the three NWP members are blended with the first three members of the nowcast (member one with member one, two with two, etc.), subsequently, the same NWP members are blended with the next three members (NWP member one with member 4, NWP member 2 with member 5, etc.), until 10 is reached.

Is this the way to go (works for me) or do we want an output of 30 (10 x 3 members here) in this case?

RubenImhoff on 9 Aug 2021 — Contributor

Oh and @wdewettin, let's add some comments to the code here so that this procedure is clear. :) (B.t.w., I can add that too, but I'll wait for the responses - and thus possible changes - of the others)

wdewettin on 9 Aug 2021 — Author

Yes! I will await the responses from the others, and then add some comments

cvelascof on 1 Oct 2021 — Member

I agree. It wuld be nice to add additional comments in this part of the script as @RubenImhoff suggested.

cvelascof on 1 Oct 2021 — Member

I agree. It would be nice to add additional comments in this part of the script as @RubenImhoff suggested.

codecov bot commented on 26 Aug 2021 · edited

Codecov Report

❗ No coverage uploaded for pull request base (`pysteps-v2@38027f8d`). Click here to learn what that means.
The diff coverage is `n/a`.

```
@@            Coverage Diff            @@
##           pysteps-v2    #229   +/-   ##
=========================================
  Coverage          ?   73.83%
=========================================
  Files             ?      152
  Lines             ?    11213
  Branches          ?        0
=========================================
  Hits              ?     8279
  Misses            ?     2934
  Partials          ?        0
```

Flag	Coverage Δ
unit_tests	73.83% <0.00%> (?)

Flags with carried forward coverage won't be shown. Click here to find out more.

Continue to review full report at Codecov.

Legend - Click here to learn more
`Δ = absolute <relative> (impact)`, `ø = not affected`, `? = missing data`
Powered by Codecov. Last update 38027f8d...75bf68b. Read the comment docs.

cvelascof reviewed on 1 Oct 2021 — View changes

pysteps/nowcasts/linear_blending.py — Outdated

```
3 + pysteps.nowcasts.linear_blending
4 + ================================
5 +
6 + Implementation of the linear blending between nowcast and NWP data.
```

cvelascof on 1 Oct 2021 — Member

@RubenImhoff @wdewettin thanks for implementing this module.

pysteps/nowcasts/linear_blending.py — Outdated

```
52 +     containing the NWP precipitation fields ordered by timestamp from oldest
53 +     to newest. The time steps between the inputs are assumed to be regular
54 +     (and identical to the time step between the nowcasts). If no NWP
55 +     data is given the value of R_nwp is None and no blending will be performed.
```

cvelascof on 1 Oct 2021 — Member

Here the object returned should be equal to the output of the nowcasting method used if R_nwp=None.
I understand that if we are going to create a function the ingest both sources of precipitation, then few lines of code must be duplicated. Perhaps in the future a higher level blending script should be created in the way that accept multiple sources of precipitation, and weights for each source (or a method to determine those weigths) and return blended fields. This approach may be easier to maintain, but perhaps harder to understand by users.

Examples of review comments

Linear blending #229

New issue

Merged RubenImhoff merged 8 commits into `pySTEPS:pysteps-v2` from `wdewettin:linear_blending` on Nov 3, 2021

Conversation 30 | Commits 8 | Checks 8 | Files changed 8 | +568 −2

wdewettin commented on Aug 9, 2021

This addresses issue #226. I added a nowcasting method called linear blending. This works for all nowcast methods (deterministic and ensemble) and is also compatible with NWP ensembles. Furthermore, I added an example which uses some dummy NWP data to blend with and which plots the results. Lastly, I also added a test for the linear blending function.

1 1

wdewettin and others added 3 commits 13 months ago

- `Implemented linear blending` 80c4d4b
- `Replaced missing data in nowcasting with NWP data` f01f4f2
- `Added compatibility for NWP ensembles` 7d6008b

Reviewers

- ladc
- cvelascof
- RubenImhoff
- dnerini

Assignees

No one assigned

Labels

enhancement

Examples of review comments

Linear blending #229

New issue

Merged RubenImhoff merged 8 commits into `pySTEPS:pysteps-v2` from `wdewettin:linear_blending` on Nov 3, 2021

Conversation 30 | Commits 8 | Checks 8 | Files changed 8 | +568 −2

wdewettin commented on Aug 9, 2021

This addresses issue #226. I added a nowcasting method called linear blending. This works for all nowcast methods (deterministic and ensemble) and is also compatible with NWP ensembles. Furthermore, I added an example which uses some dummy NWP data to blend with and which plots the results. Lastly, I also added a test for the linear blending function.

1 1

wdewettin and others added 3 commits 13 months ago

- `Implemented linear blending` ... `80c4d4b`
- `Replaced missing data in nowcasting with NWP data` ... `f01f4f2`
- `Added compatibility for NWP ensembles` `7d6008b`

Reviewers

- ladc
- cvelascof
- RubenImhoff
- dnerini

Assignees

No one assigned

Labels

enhancement

Examples of review comments

pysteps/nowcasts/linear_blending.py Outdated

```
+     )
+
+     # Transform the precipitation back to mm/h
+     R_nowcast = transformation.dB_transform(R_nowcast, threshold=-10.0, inverse=True)[0]
```

RubenImhoff on Aug 9, 2021 Contributor

I believe the threshold is present in the metadata (the transformer at least returns the metadata). We could add this to the function, then we don't have to hard code it here.

1

cvelascof on Oct 1, 2021 Member

I understand why this is done here, but this is assuming that `R_nwp` is in mm/hr and that the `precip` is in dB. If this is the case, then we need to indicate those limitations in the description of the function. Alternatively, both sources could be in `mm/hr` and the conversion of `precip` to `dB` be done within this function as well.

Said that the returned output `R_blended` will be always in `mm/hr`

I understand why this is done here, but this is assuming that `R_nwp` is in mm/hr and that the `precip` is in dB. If this is the case, then we need to indicate those limitations in the description of the function. Alternatively, both sources could be in `mm/hr` and the conversion of `precip` to `dB` be done within this function as well.

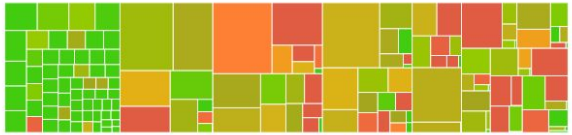
Said that the returned output `R_blended` will be always in `mm/hr`

Examples of review comments

codecov bot commented on Aug 26, 2021 • edited

Codecov Report

! No coverage uploaded for pull request base (`pysteps-v2@3802f8d`). Click here to learn what that means.
The diff coverage is `n/a`.



```
@@              Coverage Diff              @@
##             pysteps-v2     #229   +/-   ##
=============================================
  Coverage               ?   73.83%
=============================================
  Files                  ?      152
  Lines                  ?    11213
  Branches               ?        0
=============================================
  Hits                   ?     8279
  Misses                 ?     2934
  Partials               ?        0
```

Flag	Coverage Δ
unit_tests	`73.83% <0.00%> (?)`

Flags with carried forward coverage won't be shown. Click here to find out more.

Examples of review comments

RubenImhoff added 2 commits 10 months ago

`snake_case implementation and move linear blending to blending folder` `d9d0777`

`Changes after review by Carlos, Daniele and Ruben` ✗ `6d4a375`

RubenImhoff commented on Oct 19, 2021 Contributor ···

All requested changes implemented. Main task left is to make the code xarray ready for pysteps-v2. As discussed before, we may also do that in a separate PR.

`Fixes for tests` ✓ `758f88b`

RubenImhoff merged commit `d30b07e` into `pySTEPS:pysteps-v2` on Nov 3, 2021 View details

8 checks passed

Git flow

1. **Create a branch**
2. **Make changes/improvements/developments**
3. **Create a pull request**
4. **Address review comments**
5. **Merge your pull request**

Once your pull request is approved by the pysteps team, they merge your pull request.

This will automatically merge your branch so that your changes appear on the main branch.

Git flow

1. **Create a branch**
2. **Make changes/improvements/developments**
3. **Create a pull request**
4. **Address review comments**
5. **Merge your pull request**
6. **Delete your branch**

After you merge your pull request, delete your branch. This indicates that the work on the branch is complete and prevents you or others from accidentally using old branches.

Best practices: Naming conventions

- ➔ Follow the conventions.
- ➔ Use concise, self-explanatory names.
- ➔ Avoid mathematical syntax («X», «Y», «Z»).
- ➔ The length of a variable name should reflect its importance.
- ➔ The name of a function should tell what it does.
- ➔ Beware of typos and grammar, use a spell checker.

Type	Convention
Packages	lower_with_under
Module	lower_with_under
Classes	CapWords
Exceptions	CapWords
Functions	lower_with_under()
Global/Class Constants	CAPS_WITH_UNDER
Global/Class Variables	lower_with_under
Instance Variables	lower_with_under
Method Names	lower_with_under()
Function/Method Parameters	lower_with_under
Local Variables	lower_with_under

Best practices: Code styling

Follow Python's code style guidelines (PEP8).

Use an auto-formatter (black is pre-committed in pySTEPS).

- Avoid extraneous whitespace
- Max line-length: 88 characters
- Always indent wrapped code for readability

Always use four spaces for indentation (don't use tabs).

Avoid writing multiple statements in the same line.

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Best practices: Readability/Modularity

Avoid long scripts by organizing your code into functions. Each function should represent only one action.

```python
import random
import sys

##############################################
# parse input
num = sys.argv[1]

##############################################
# generate random numbers
rand_num = [random.random() for _ in range(num)]

##############################################
# add all random numbers together
total = 0
for number in random_numbers:
  total += number

print(f"Total={total}")
```

```python
import random
import sys

def get_random_numbers(num):
  return [random.random() for _ in range(num)]

def take_sum(numbers):
  total = 0
  for number in numbers:
    total += number
  return total

if __name__ == "__main__":
  numbers = get_random_numbers(sys.argv[1])
  total = take_sum(numbers)
  print(f"Total={total}")
```

Test the code

- Create a minimal test routine for your code change.
- e.g., if you fixed a bug, implement a new test that can reproduce that same bug.
- The tests should be placed under the [pysteps.tests]() module.
- The file should follow the **test_*.py** naming convention and have a descriptive name.

```python
def get_random_numbers(num_random):
    """
    Get random numbers.

    Parameters
    ----------
    num_random : int
        The number of random numbers.

    Returns
    -------
    output : list
        List of random numbers.

    """
    return [random.random() for _ in range(num_random)]


def test_get_random_numbers():
    """Test the correctness of get_random_numbers().
    """
    assert len(get_random_numbers(3)) == 3
```

Test your changes

- Tests are organized in a dedicated folder in your project.
- Run the whole test suite with pytest to make sure that your changes didn't break anything elsewhere.

```
(pysteps_dev) bash-4.2$ pytest pysteps/tests/ --disable-warnings
========================================= test session starts =========================================
platform linux -- Python 3.9.4, pytest-6.2.4, py-1.10.0, pluggy-0.13.1
rootdir: /users/ned/pySTEPS/pysteps
plugins: cov-2.11.1
collected 443 items / 1 skipped / 442 selected

pysteps/tests/test_archive.py ......                                                          [  1%]
pysteps/tests/test_cascade.py .....                                                           [  2%]
pysteps/tests/test_datasets.py ...........                                                    [  4%]
pysteps/tests/test_detcatscores.py ......................                                     [  9%]
pysteps/tests/test_detcontscores.py ..............................                            [ 16%]
pysteps/tests/test_downscaling_rainfarm.py .....                                              [ 17%]
pysteps/tests/test_ensemblestats.py ......................                                    [ 22%]
pysteps/tests/test_ensscores.py ........                                                      [ 23%]
pysteps/tests/test_exporters.py ..                                                            [ 24%]
pysteps/tests/test_extrapolation_semilagrangian.py .                                          [ 24%]
pysteps/tests/test_feature_tstorm.py ...                                                      [ 25%]
pysteps/tests/test_importer_decorator.py ......                                               [ 26%]
pysteps/tests/test_interfaces.py ..........                                                   [ 28%]
pysteps/tests/test_io_bom_rf3.py ............................                                 [ 34%]
pysteps/tests/test_io_fmi_pgm.py ............................                                 [ 41%]
pysteps/tests/test_io_knmi_hdf5.py ..................                                         [ 45%]
pysteps/tests/test_io_mch_gif.py ............................                                 [ 51%]
pysteps/tests/test_io_mrms_grib.py .                                                          [ 51%]
pysteps/tests/test_io_opera_hdf5.py ....................                                      [ 56%]
pysteps/tests/test_io_saf_crri.py ................                                            [ 59%]
pysteps/tests/test_motion.py ......FF...............                                          [ 65%]
pysteps/tests/test_motion_lk.py .......                                                       [ 66%]
pysteps/tests/test_noise_motion.py ..........                                                 [ 68%]
pysteps/tests/test_nowcasts_anvil.py ..                                                       [ 69%]
pysteps/tests/test_nowcasts_sprog.py ......                                                   [ 70%]
pysteps/tests/test_nowcasts_sseps.py ..                                                       [ 71%]
pysteps/tests/test_nowcasts_steps.py .........                                                [ 73%]
pysteps/tests/test_paramsrc.py .                                                              [ 73%]
pysteps/tests/test_plt_animate.py .                                                           [ 73%]
pysteps/tests/test_plt_cartopy.py ..........                                                  [ 75%]
pysteps/tests/test_plt_motionfields.py .......                                                [ 77%]
pysteps/tests/test_plt_precipfields.py ...........                                            [ 79%]
pysteps/tests/test_probscores.py ...                                                          [ 80%]
pysteps/tests/test_spatialscores.py ....                                                      [ 81%]
pysteps/tests/test_timeseries_autoregression.py ............                                  [ 84%]
pysteps/tests/test_tracking_tdating.py ...                                                    [ 84%]
pysteps/tests/test_utils_arrays.py ....                                                       [ 85%]
pysteps/tests/test_utils_cleansing.py ........                                                [ 87%]
pysteps/tests/test_utils_conversion.py ...........................                            [ 93%]
pysteps/tests/test_utils_dimension.py ...................                                     [ 97%]
pysteps/tests/test_utils_transformation.py .........                                          [100%]
================= 2 failed, 441 passed, 1 skipped, 105 warnings in 241.42s (0:04:01) =================
```

Examples: documentation

Document all your modules, class, and functions (PEP257).

A summary line, followed by a single blank line, followed by a more elaborate description.

Use the Numpy's docstring format.

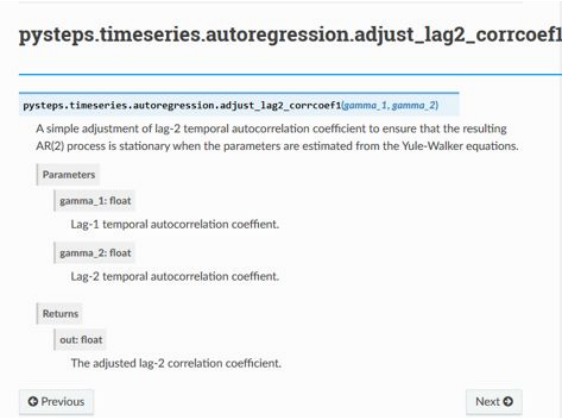
Can be used by automatic documentation tools such as Sphinx.

Add usage example.

```python
def adjust_lag2_corrcoef1(gamma_1, gamma_2):
    """A simple adjustment of lag-2 temporal autocorrelation coefficient to
    ensure that the resulting AR(2) process is stationary when the parameters
    are estimated from the Yule-Walker equations.

    Parameters
    ----------
    gamma_1 : float
      Lag-1 temporal autocorrelation coeffient.
    gamma_2 : float
      Lag-2 temporal autocorrelation coeffient.

    Returns
    -------
    out : float
      The adjusted lag-2 correlation coefficient.
    """
```



Minor issues/improvements for the hands-on activities

1. Add tests
    - Issue 1: Test that a ValueError exception is raised when the input array contains NaNs
    - Issue 2: Test that input arrays with more than 2 dimensions raise a ValueError exception
    - Issue 3: Test that a 1D-array with the coordinates' scale is accepted
    - Issue 4: Test that a ValueError exception is raised when the input array contains NaNs
    - Issue 5: Test that input arrays with more than 2 dimensions raise a ValueError exception
    - Issue 6: Test that input arrays with incorrect dimensions raise a ValueError exception
    - Issue 7: Test that passing the timesteps raises a ValueError exception if they are not sorted in ascending order
2. Small code-refactorings
    - Issue 8: Improve the get_method interface from the pysteps.verification.interface.py module
    - Issue 9: Replace collections.abc.Iterable by an explicit import of the Iterable class in detcatscores module
    - Issue 10: Replace collections.abc.Iterable by an explicit import of the Iterable class in the detcontscores module
3. Replace old style string formatting (% Operator) with f-strings
    - Issue 11: Replace old style string formatting (% operator) with f-strings in pysteps.utils.tapering module
    - Issue 12: Replace old style string formatting (% operator) with f-strings in pysteps.blending.steps module
    - Issue 13: Replace old style string formatting (% operator) with f-strings in pysteps.utils.spectral module
    - Issue 14: Replace old style string formatting (% operator) with f-strings in pysteps.io.archive module
    - Issue 15: Replace old style string formatting (% operator) with f-strings in pysteps.utils.cleansing module