Bidirectional Reach (BReach) Documentation

Release 0.2.0

Stijn Van Hoey

Apr 25, 2020

Contents

1 Bidirectional Reach (BReach) 3
- 1.1 Features . . . 3
- 1.2 Input . . . 4
- 1.3 Ouput . . . 4
- 1.4 Credits . . . 4

2 Installation 5
- 2.1 Stable release . . . 5
- 2.2 From sources . . . 5

3 Usage 7

4 Contributing 9
- 4.1 Types of Contributions . . . 9
- 4.2 Get Started! . . . 10
- 4.3 Pull Request Guidelines . . . 11
- 4.4 Tips . . . 11

5 Indices and tables 13

Contents:

CHAPTER 1

Bidirectional Reach (BReach)

This Python Package provides an implementation of the methodology presented in Identification of temporal consistency in rating curve data: Bidirectional Reach (BReach). BReach identifies the consistency of rating curve data based on a quality analysis of model results. Results of this analysis enable the detection of changes in data consistency.

- Free software: MIT license
- Documentation: https://pybreach.readthedocs.io.

1.1 Features

The BReach methodology consists of different steps, that are described in Van Eerdenbrugh et al., 2016 :

- Step 1: Selection of a model structure for the analysis;
- Step 2: Sampling of the parameter space;
- Step 3: Assessment of acceptable model results;
- Step 4: Assessment of different degrees of tolerance;
- Step 5: Assessment of the bidirectional reach for all degrees of tolerance;
- Step 6: Identification of consistent data periods.

The current scope of the package is to support the execution of steps 4, 5 and 6. Based on a two-dimensional matrix of performance measures (calculated for each data point and a given number of parameter sets), the package calculates the maximum left and right reaches in each data point for different degrees of tolerance and provides the visualisation(s) to interpret the data.

1.2 Input

A user has thus to prepare steps 1 - 3 of the methodology prior to the use of the pybreach package. Inputs for the package are:

- A two-dimensional matrix (numpy ndarray) of shape NxM with N the number of model realisations and M the number of model evaluation points (time steps, measured values). This matrix contains binary information that results from step 3 (value '1' = acceptable model result, value '0' = nonacceptable model result).
- A list containing different degrees of tolerance, defining the percentage of points that are allowed to be nonacceptable in the BReach analysis. In both Van Eerdenbrugh et al., 2016 and Van Eerdenbrugh et al., 2017, degrees of tolerance of 0 %, 5 %, 10 %, 20 % and 40 % are used.

1.3 Ouput

The script pybreach.py calculates the maximum left and right reach for a given matrix with model evaluations. The result is a numpy ndarray of shape NxM in which N is the number of model evaluation points and M is 2 * the amount of degrees of tolerance. Columns (2*i-1) contain the maximum left reaches and columns 2*i contain the maximum right reaches for all data points and degree of tolerance i.

The script breachplot.py returns a BReach plot for a given BReach result.

1.4 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install Bidirectional Reach (BReach), run this command in your terminal:

```
$ pip install pybreach
```

This is the preferred method to install Bidirectional Reach (BReach), as it will always install the most recent stable release.

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

2.2 From sources

The sources for Bidirectional Reach (BReach) can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/stijnvanhoey/pybreach
```

Or download the tarball:

```
$ curl  -OL https://github.com/stijnvanhoey/pybreach/tarball/master
```

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

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

CHAPTER 3

Usage

To use Bidirectional Reach (BReach) in a project:

```
from pybreach.pybreach import breach_run
```

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/stijnvanhoey/pybreach/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/stijnvanhoey/pybreach/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 :)

4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/pybreach.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 pybreach
$ cd pybreach/
$ 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 pybreach tests
$ python setup.py test or py.test
$ 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.

4.3 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 2.7, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/stijnvanhoey/pybreach/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_pybreach
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search