wwdata Documentation

Release 0.2.0

Chaim De Mulder

Apr 10, 2019

Contents

Contents:

CHAPTER 1

wwdata

Data analysis package aimed at data obtained in the context of (waste)water

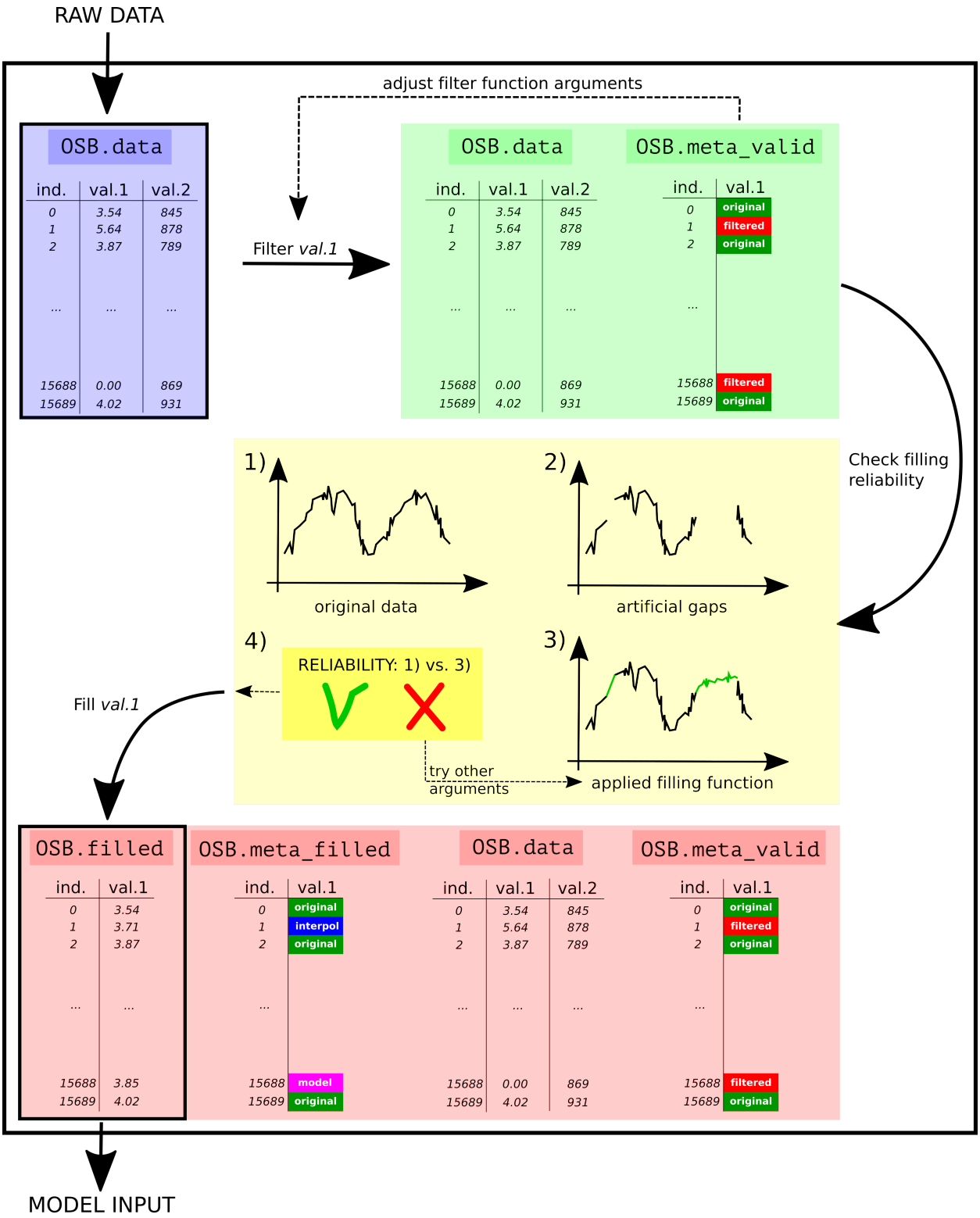
- Free software: GNU General Public License v3
- Documentation: https://ugentbiomath.github.io/wwdata-docs/
- Funding: Waterboard De Dommel
- Context: PhD research at BIOMATH, Ghent University

1.1 Structure

The package contains one class and three subclasses, all in separate .py files. Division in subclasses is based on the type of data:

- online data from full scale installations (OnlineSensorBased)
- online data from lab experiments (LabSensorBased)
- offline data obtained from lab experiments (LabExperimentBased).

Jupyter notbeook files (.ipynb) illustrate the use of the available functions. The most developed class is the OnlineSensorBased one. The workflow of this class is shown in below Figure, where OSB represents an OnlineSensorBased object. Main premises are to never delete data but to tag it and to be able to check the reliability when gaps in datasets are filled.


RAW DATA
adjust filter function arguments
OSB.data
ind. val.1 val.2
0 3.54 845
1 5.64 878
2 3.87 789
... ...
15688 0.00 869
15689 4.02 931
Filter val.1
OSB.data
OSB.meta_valid
ind. val.1 val.2
0 3.54 845
1 5.64 878
2 3.87 789
... ...
15688 0.00 869
15689 4.02 931
ind. val.1
0 original
1 filtered
2 original
...
15688 filtered
15689 original
Check filling reliability
1)
original data
2)
artificial gaps
4)
RELIABILITY: 1) vs. 3)
3)
applied filling function
try other arguments
Fill val.1
OSB.filled
ind. val.1
0 3.54
1 3.71
2 3.87
... ...
15688 3.85
15689 4.02
OSB.meta_filled
ind. val.1
0 original
1 interpol
2 original
...
15688 model
15689 original
OSB.data
ind. val.1 val.2
0 3.54 845
1 5.64 878
2 3.87 789
... ...
15688 0.00 869
15689 4.02 931
OSB.meta_valid
ind. val.1
0 original
1 filtered
2 original
...
15688 filtered
15689 original
MODEL INPUT

1.2 Examples

For the workflow with code and more specific examples, check out the Showcase Jupyter Notebook(s) included as documentation of the package.

1.3 Credits

This package was created with support from Cookiecutter and the audreyr/cookiecutter-pypackage project template, as well as this GitHub page, provided by Daler and explaining how to use sphinx documentation generation in combination with GitHub Pages.

CHAPTER 2

Installation

2.1 Stable release

To install wwdata, run this command in your terminal:

```
$ pip install wwdata
```

This is the preferred method to install wwdata, 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 wwdata can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/cdemulde/wwdata
```

Or download the tarball:

```
$ curl  -OL https://github.com/cdemulde/wwdata/tarball/master
```

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

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

CHAPTER 3

Usage

To use wwdata in a project:

```
import wwdata as ww
```

As the package was developed with the help of Jupyter Notebook (version 4.4.1) and visualisation is an important part of it, it is highly recommended to make use of a Jupyter Notebook when using the package.

Once the package is imported, the suggested workflow is as follows:

1. Read data and convert it to a pandas DataFrame (see [the pandas documentation](https://pandas.pydata.org) for more information); format it in the way you like.
2. Create an object of any of the three classes in the wwdata package, e.g. the OnlineSensorBased class:

   ```
   data = ww.OnlineSensorBased(dataframe,timedata_column="time",data_type="WWTP",
   ↪experiment_tag="Data 2017")
   ```

3. Explore and format the data (convert to datetime, make absolute time index, make some plots. . .), e.g.:

   ```
   data.to_datetime(time_column="time",time_format="%dd-%mm-%yy")
   data.get_avg()
   ```

4. Tag non-valid data points. The way to do this depends on the data you are working with, but the general approach would be:

   ```
   data.tag_nan()
   data.moving_slope_filter(xdata="time",data_name="series1",cutoff=3,arange=['1/1/
   ↪2017','1/2/2017'])
   ```

i.e. to simply apply any of the filtering functions to the class object.

5. Apply any other functionalities to the object. In the below example, this is the filling of the gaps introduced by filtering data in the previous step (for details on the meaning of the arguments, please refer to the documentation provided within the source code):

```
data.fill_missing_interpolation("series1",range_=12,arange=['1/1/2017','1/2/2017
↪'])
data.fill_missing_model("series1","model_data_series",arange=['1/1/2017','1/2/2017
↪'])
```

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/cdemulde/wwdata/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

wwdata could always use more documentation, whether as part of the official wwdata 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/cdemulde/wwdata/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 *wwdata* for local development.

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

```
$ git clone git@github.com:your_name_here/wwdata.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 wwdata
$ cd wwdata/
$ 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 wwdata 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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/cdemulde/wwdata/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_wwdata
```

CHAPTER 5

Credits

5.1 Development Lead

Chaim De Mulder

5.2 Contributors

None yet. All contributions are welcome!

5.3 Current funder

Waterboard De Dommel (The Netherlands)

CHAPTER 6

History

6.1 0.1.0 (2017-10-23)

First release on PyPI.

The wwdata (wastewater data) package is meant to make data analysis, validation and filling of data gaps more streamlined. It contains code to do all this, while also providing simple visualisations of the whole procedure.

The package was (and is) developed in the framework of PhD research, involving the modelling of a full scale wastewater treatment plant (WWTP). Online measurements at the plant are available, but as with all data, is not perfect and therefor needs validation. The gap filling originated from the need to have high-frequency influent data available to run the WWTP model with.

6.2 0.2.0 (2018-06-12)

Second release on PyPI.

The wwdata (wastewater data) package is meant to make data analysis, validation and filling of data gaps more streamlined. It contains code to do all this, while also providing simple visualisations of the whole procedure.

The package was (and is) developed in the framework of PhD research, involving the modelling of a full scale wastewater treatment plant (WWTP). Online measurements at the plant are available, but as with all data, is not perfect and therefor needs validation. The gap filling originated from the need to have high-frequency influent data available to run the WWTP model with.

New in version 0.2.0:

- Bug fixes
- Addition of an `only_checked` argument to multiple functions to allow application of the function to only the validated data points ('original' in self.meta_valid).
- Extended, improved and customized documentation website (generated with sphinx).
- Extended and improved Jupyter Notebook for documentation.

- Improved visualisation for *get_correlation*: a prediction band based on the obtained correlation is now included in the produced scatter plot.

Known bugs:

- See [open issues on Github](https://github.com/UGentBiomath/wwdata/issues)

CHAPTER 7

Indices and tables

- genindex
- modindex
- search