https://github.com/computationalmodelling/nbval

This repository | Search | Pull requests | Issues | Gist

computationalmodelling / **nbval**

Watch 7 | Star 41 | Fork 10

Code | Issues 4 | Pull requests 1 | Projects 0 | Wiki | Pulse | Graphs

A py.test plugin to validate Jupyter notebooks

ipython-notebook | jupyter-notebook | python | testing | pytest-plugin | pytest

273 commits | **2** branches | **6** releases | **9** contributors

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

takluyver Version number -> 0.5 — Latest commit `1997b75` 3 days ago

issues	Adding example file for issue #27	a month ago
nbval	Version number -> 0.5	3 days ago
tests	Timeouts test should not capture	14 days ago
utils	Updated documentation. Organised files. Clean.	2 years ago
.gitignore	Ignore py.test .cache folder	20 days ago
.hgignore	Updated ignore file	2 years ago
.travis.yml	Have travis update pip/setuptools	19 days ago
CONTRIBUTORS	authors	a year ago
INSTALL	Added installation instructions and cleaned up the main file.	2 years ago
LICENSE	Update URL, author details.	2 years ago
Makefile	I like the green "PASSED"	14 days ago
README.md	Add spaces in copy and pasted ODK reference	a month ago
doc_sanitize	Updated test notebook to include examples and instructions for usage	2 years ago
documentation.ipynb	Also document tags	18 days ago
setup.cfg	Lint	24 days ago
setup.py	Version number -> 0.5	3 days ago

README.md

Py.test plugin for validating Jupyter notebooks

build passing

The plugin adds functionality to py.test to recognise and collect Jupyter notebooks. The intended purpose of the tests is to determine whether execution of the stored inputs match the stored outputs of the `.ipynb` file. Whilst also ensuring that the notebooks are running without errors.

The tests were designed to ensure that Jupyter notebooks (especially those for reference and documentation), are executing consistently.

Each cell is taken as a test, a cell that doesn't reproduce the expected output will fail.

See `documentation.ipynb` for the full documentation.

Installation

Available on PyPi:

```
pip install nbval
```

or install the latest version from cloning the repository and running:

```
pip install .
```

from the main directory. To uninstall:

```
pip uninstall nbval
```

How it works

The extension looks through every cell that contains code in an IPython notebook and then the `py.test` system compares the outputs stored in the notebook with the outputs of the cells when they are executed. Thus, the notebook itself is used as a testing function. The output lines when executing the notebook can be sanitized passing an extra option and file, when calling the `py.test` command. This file is a usual configuration file for the `ConfigParser` library.

Regarding the execution, roughly, the script initiates an IPython Kernel with a `shell` and an `iopub` sockets. The `shell` is needed to execute the cells in the notebook (it sends requests to the Kernel) and the `iopub` provides an interface to get the messages from the outputs. The contents of the messages obtained from the Kernel are organised in dictionaries with different information, such as time stamps of executions, cell data types, cell types, the status of the Kernel, username, etc.

In general, the functionality of the IPython notebook system is quite complex, but a detailed explanation of the messages and how the system works, can be found here

http://ipython.org/ipython-doc/stable/development/messaging.html

Execution

To execute this plugin, you need to execute `py.test` with the `nbval` flag to differentiate the testing from the usual python files:

```
py.test --nbval
```

You can also specify `--nbval-lax`, which runs notebooks and checks for errors, but only compares the outputs of cells with a `#NBVAL_CHECK_OUTPUT` marker comment.

```
py.test --nbval-lax
```

The commands above will execute all the `.ipynb` files in the current folder. Alternatively, you can execute a specific notebook:

```
py.test --nbval my_notebook.ipynb
```

If the output lines are going to be sanitized, an extra flag, `--sanitize-with` together with the path to a confguration file with regex expressions, must be passed, i.e.

```
py.test --nbval my_notebook.ipynb --sanitize-with path/to/my_sanitize_file
```

where `my_sanitize_file` has the following structure.

```
[Section1]
regex: [a-z]*
replace: abcd

regex: [1-9]*
replace: 0000

[Section2]
regex: foo
```

```
replace: bar
```

The `regex` option contains the expression that is going to be matched in the outputs, and `replace` is the string that will replace the `regex` match. Currently, the section names do not have any meaning or influence in the testing system, it will take all the sections and replace the corresponding options.

Help

The `py.test` system help can be obtained with `py.test -h`, which will show all the flags that can be passed to the command, such as the verbose `-v` option. The IPython notebook plugin can be found under the `general` section.

Acknowledgements

This plugin was inspired by Andrea Zonca's py.test plugin for collecting unit tests in the IPython notebooks (https://github.com/zonca/pytest-ipynb).

The original prototype was based on the template in https://gist.github.com/timo/2621679 and the code of a testing system for notebooks https://gist.github.com/minrk/2620735 which we integrated and mixed with the `py.test` system.

We acknowledge financial support from

- OpenDreamKit Horizon 2020 European Research Infrastructures project (#676541), http://opendreamkit.org
- EPSRC's Centre for Doctoral Training in Next Generation Computational Modelling, http://ngcm.soton.ac.uk (#EP/L015382/1) and EPSRC's Doctoral Training Centre in Complex System Simulation ((EP/G03690X/1),
- The Gordon and Betty Moore Foundation through Grant GBMF #4856,by the Alfred P. Sloan Foundation and by the Helmsley Trust.

Authors

2014 - 2017 David Cortes-Ortuno, Oliver Laslett, T. Kluyver, Vidar Fauske, Maximilian Albert, MinRK, Ondrej Hovorka, Hans Fangohr