NBVAL: use case and introduction

Hans Fangohr, Marijan Beg, Vidar Tonaas Fauske, Thomas Kluyver and others

Jupyter Workshop, Edinburgh, January 2017

Overview – NBVAL

- Notebook use cases
- Automation in Software Engineering as motivation
- NBVAL
- Demos
- Upcoming work and request for input
- Summary

Jupyter Notebook use cases

- Research: Notebooks for computational exploration
    - documentation of computational study
    - reproducibility
    - collaborative features
    - dissemination
- Software Engineering: Notebooks for documentation:
    - documenting software with the Notebook
    - Tutorials
    - Walk-throughs
    - Example studies

Software Engineering: Automate everything

Automate

- unit, system, regression tests
- building of
  - binaries and distribution files
  - documentation
- Often called ”Continuous integration (CI)”, popular services:
  - Travis CI, Cirlce CI and others
  - Jenkins, BuildBot, …

NBVAL

- automate the *validation* of notebooks used

Prerequisites

- Jupyter Notebook
- `py.test`

NoteBook VALidation (NBVAL)

NBVAL

NBVAL validates a saved notebook in the sense that stored input cells produce output cells that are identical to the output cell data saved in the notebook.

Typical work flow:

- create notebook (making use of software via `import` commands)
- execute cells and save notebook with output
- run `nbval` in the future to *validate* notebook

Use cases:

- automatically check that documentation is correct
- increase test coverage

Installation

```
$ conda create -n nbval python=3 notebook pytest
$ pip install nbval
```

Check that the nbval plugin is installed:

```
(nbval) $ py.test --version
This is pytest version 3.0.5, imported from /Users/fangohr/ana
    envs/nbval/lib/python3.6/site-packages/pytest.py
setuptools registered plugins:
  nbval-0.3.6 at /Users/fangohr/anaconda3/envs/nbval/lib/pytho
    site-packages/nbval/plugin.py
(nbval) $
```

Example: Demo 1 - basics

- `py.test -v --nbval demo1.ipynb`
- `# NBVAL_IGNORE_OUTPUT`

Example: Demo 2 - dates and times; sanitize

- `py.test -v --nbval demo2.ipynb`
- `py.test --nbval -v demo2.ipynb`
  `--sanitize-with sanitize_demo2.cfg`
- `sanitize_demo2.cfg`:

```
[regex1]
regex: \d{1,2}/\d{1,2}/\d{2,4}
replace: DATE-STAMP

[regex2]
regex: \d{2}:\d{2}:\d{2}
replace: TIME-STAMP
```

Example: Demo 3 - matplotlib / memory address

- `py.test -v --nbval demo3.ipynb`
- `py.test -v --nbval`

  `--sanitize-with sanitize_mem.cfg demo3.ipynb`
- `sanitize\_mem.cfg`:

```
[Memory addresses]
regex: 0x[0-9a-fA-F]+
replace: MEMORY_ADDRESS
```

Example: Demo 4 `--nbval-lax`

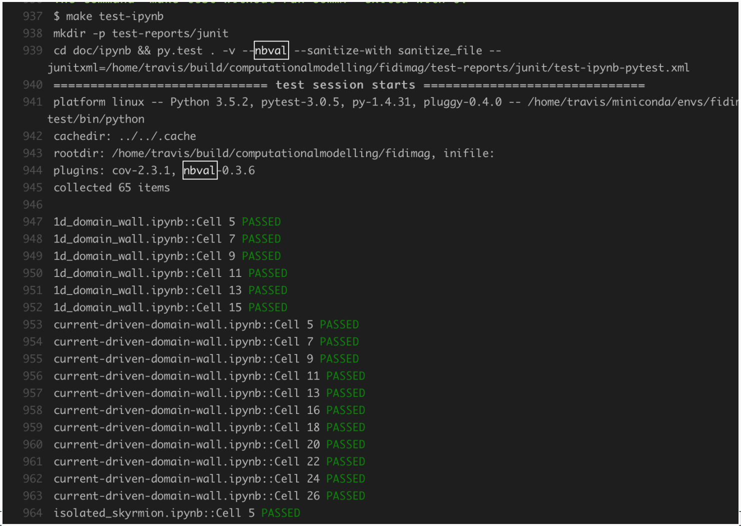
Alternative use: more reLAXed approach

`--nbval`	`--nbval-lax`
output must agree	output can differ
no exceptions raised	no exceptions raised

- When using `lax` mode, we can fore checking out put with `#NBVAL_CHECK_OUTPUT`
- Useful to make use of existing notebooks immediately

Example use on Travis

https://travis-ci.org/computationalmodelling/fidimag/jobs/187178465



```
$ make test-ipynb
mkdir -p test-reports/junit
cd doc/ipynb && py.test . -v --nbval --sanitize-with sanitize_file --
junitxml=/home/travis/build/computationalmodelling/fidimag/test-reports/junit/test-ipynb-pytest.xml
============================ test session starts ==============================
platform linux -- Python 3.5.2, pytest-3.0.5, py-1.4.31, pluggy-0.4.0 -- /home/travis/miniconda/envs/fidi
test/bin/python
cachedir: ../../.cache
rootdir: /home/travis/build/computationalmodelling/fidimag, inifile:
plugins: cov-2.3.1, nbval-0.3.6
collected 65 items

1d_domain_wall.ipynb::Cell 5 PASSED
1d_domain_wall.ipynb::Cell 7 PASSED
1d_domain_wall.ipynb::Cell 9 PASSED
1d_domain_wall.ipynb::Cell 11 PASSED
1d_domain_wall.ipynb::Cell 13 PASSED
1d_domain_wall.ipynb::Cell 15 PASSED
current-driven-domain-wall.ipynb::Cell 5 PASSED
current-driven-domain-wall.ipynb::Cell 7 PASSED
current-driven-domain-wall.ipynb::Cell 9 PASSED
current-driven-domain-wall.ipynb::Cell 11 PASSED
current-driven-domain-wall.ipynb::Cell 13 PASSED
current-driven-domain-wall.ipynb::Cell 16 PASSED
current-driven-domain-wall.ipynb::Cell 18 PASSED
current-driven-domain-wall.ipynb::Cell 20 PASSED
current-driven-domain-wall.ipynb::Cell 22 PASSED
current-driven-domain-wall.ipynb::Cell 24 PASSED
current-driven-domain-wall.ipynb::Cell 26 PASSED
isolated_skyrmion.ipynb::Cell 5 PASSED
```

Feature wish list

- autocompletion
- nbdiff output on error / for selected cell?
- debug output after sanitising
- connect to ‘coverage‘ tool to record code coverage from ipynb-“tests”
- ...

Summary

NBVAL

- Validate saved notebook:
- Re-execute code cell and compare
    - computed output with
    - stored output
- report test failure if outputs disagree (`--nbval`)
- report test failure if exception is raised (`--nbval-lax`)

Project home page

- github.com/computationalmodelling/nbval

Acknowledgements

Contributors:

David Cortes-Ortuno, Oliver Laslett, Vidar Tonaas Fauske, Thomas Kluyver, Maximilian Albert, Marijan Beg, Ondrej Hovorka, Hans Fangohr

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.