Maintaining your code and tests

Ellert van der Velden

ADACS Code Testing Workshop 2019

Why should you maintain your code (and tests)?

- Common responses against:
  - I am the only one using my code (also in response to documentation);
  - It works on my machine!;
  - I know how to use my code;
  - My tests are sufficient enough;
  - Etc…

Why should you maintain your code (and tests)?

- My counter-responses:
  - Python is a very rapidly evolving programming language:
    - ➢ No single minor version has survived for 2 years before being succeeded.
  - Python is also increasingly becoming more popular:
    - ➢ Currently the 3rd most popular programming language, after C and Java.
  - Future you is a different person.
- Python code/packages become outdated incredibly quickly.

Python basics

- Python is designed as an open-source programming language;
- Code sharing and recycling is encouraged;
- Result: Your code probably relies on a few (or more) third-party packages;
- Solution: Write these down in a *requirements.txt* file;
    - Setup files can take these requirements into account automatically.

Code requirements

- Two common mistakes:
  - Not specifying all requirements;
  - Not specifying minimum required versions.
- Both are annoying and tedious to deal with as a user, especially the latter.

Not specifying all code requirements

- Common reasons:
  - The requirement in question is a very common package, like NumPy;
  - The requirement in question is satisfied by another requirement.
- This is wrong as you cannot guarantee these assumptions.

- TODO: Specify all YOUR imports as requirements, unless they are builtins.

Not specifying minimum required versions

- Specifying the minimum required version guarantees functionality;
- Failing to do so can lead to irritating and frustrating situations for the user.

- TODO: Use your current versions as the minimum versions.

Pytest plugins

- Many packages provide plugins for *pytest* to make it easier to test certain features of your code:
  - *pytest-mpl*: Tools for testing and comparing Matplotlib figures;
  - *pytest-pep8*: Tests if your code is PEP8-compliant;
  - *pytest-cov*: Check the code coverage of your tests.
- Can be easily installed using *pip* and enabled with *pytest --xxx*.

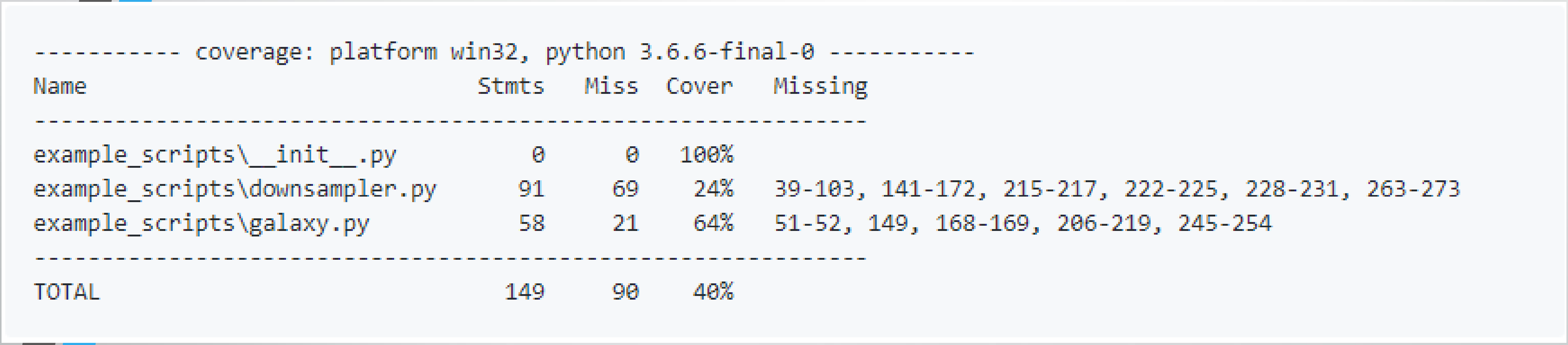
Code coverage: Why?

- Write near-exhaustive tests;
- Check for code redundancy;
- Find non-covered code;
- Special test-cases help in the future.

Code coverage: How?

- Aim for 100% coverage, including branch coverage;
- If that is not possible, ask yourself why (maybe use *pragma: no cover*);
- Make sure to write a single test for a single coverage case (e.g., do not cover multiple exception cases in the same test).

Code coverage: What?

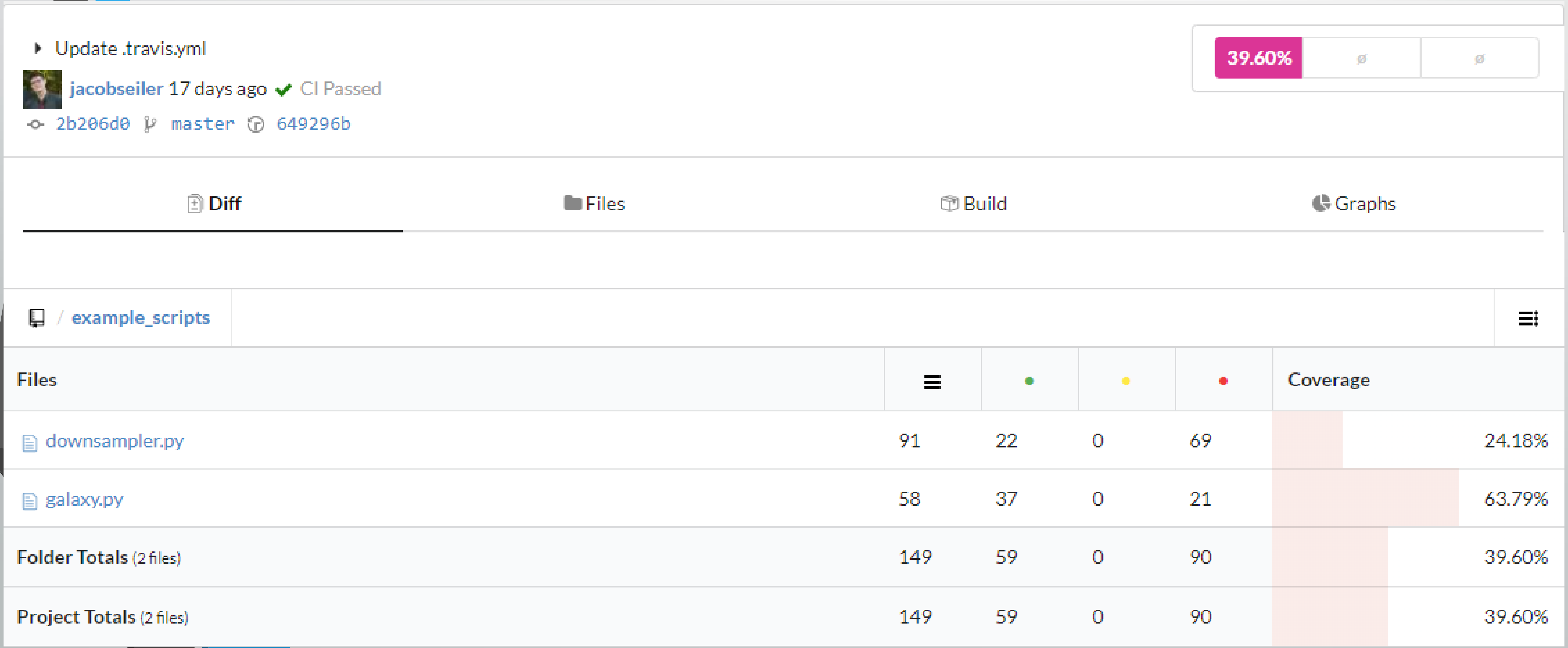


```
----------- coverage: platform win32, python 3.6.6-final-0 -----------
Name                              Stmts   Miss  Cover   Missing
--------------------------------------------------------------
example_scripts\__init__.py           0      0   100%
example_scripts\downsampler.py       91     69    24%   39-103, 141-172, 215-217, 222-225, 228-231, 263-273
example_scripts\galaxy.py            58     21    64%   51-52, 149, 168-169, 206-219, 245-254
--------------------------------------------------------------
TOTAL                               149     90    40%
```

Code coverage: Where?

- Once you have your coverage reports, you can upload them to CodeCov;
- CodeCov keeps track of your code coverage;
- It can also provide commit status messages;
- When using CI services like Travis CI, this process can be automated.

Code coverage: Where?



Update .travis.yml

jacobseiler 17 days ago ✓ CI Passed

2b206d0 master 649296b

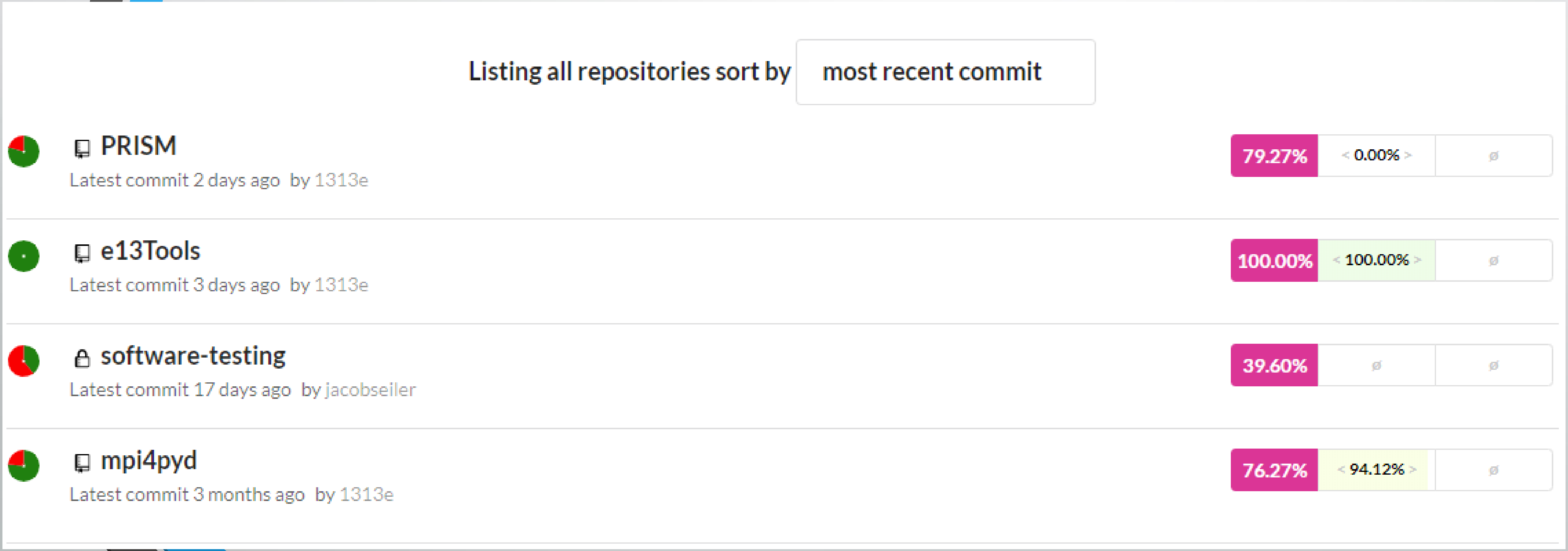
39.60% ø ø

Diff | Files | Build | Graphs

/ example_scripts

Files	≡	●	●	●	Coverage
downsampler.py	91	22	0	69	24.18%
galaxy.py	58	37	0	21	63.79%
Folder Totals (2 files)	149	59	0	90	39.60%
Project Totals (2 files)	149	59	0	90	39.60%

Code coverage: Where?



Listing all repositories sort by most recent commit

PRISM Latest commit 2 days ago by 1313e	79.27%	< 0.00% >	ø
e13Tools Latest commit 3 days ago by 1313e	100.00%	< 100.00% >	ø
software-testing Latest commit 17 days ago by jacobseiler	39.60%	ø	ø
mpi4pyd Latest commit 3 months ago by 1313e	76.27%	< 94.12% >	ø

Improving your maintenance

- Maintaining your code makes everybody happier;
- Specify all your requirements with minimum versions;
- Aim for 100% code coverage.

- Use a CI service to automate most of the process…

Coverage TL;DR

- “*pip install pytest-cov*”;
- Add arguments “*--cov --cov-report=term-missing*” to pytest (or to “*addopts*”);
- Optionally add “*--cov-branch*” for branch coverage;
- Activate repo on CodeCov to start monitoring coverage.