pytest-cov

Release 2.6.1

Mar 15, 2019

Contents

Contents:

Overview

docs	
tests	
package	

This plugin produces coverage reports. Compared to just using `coverage run` this plugin does some extras:

- Subprocess support: you can fork or run stuff in a subprocess and will get covered without any fuss.

- Xdist support: you can use all of pytest-xdist's features and still get coverage.

- Consistent pytest behavior. If you run `coverage run -m pytest` you will have slightly different `sys.path` (CWD will be in it, unlike when running `pytest`).

All features offered by the coverage package should work, either through pytest-cov's command line options or through coverage's config file.

- Free software: MIT license

1.1 Installation

Install with pip:

```
pip install pytest-cov
```

For distributed testing support install pytest-xdist:

```
pip install pytest-xdist
```

1.1.1 Upgrading from ancient pytest-cov

pytest-cov 2.0 is using a new `.pth` file (`pytest-cov.pth`). You may want to manually remove the older `init_cov_core.pth` from site-packages as it's not automatically removed.

1.1.2 Uninstalling

Uninstall with pip:

```
pip uninstall pytest-cov
```

Under certain scenarios a stray `.pth` file may be left around in site-packages.

- *pytest-cov 2.0* may leave a `pytest-cov.pth` if you installed without wheels (`easy_install`, `setup.py install` etc).

- *pytest-cov 1.8 or older* will leave a `init_cov_core.pth`.

1.2 Usage

```
pytest --cov=myproj tests/
```

Would produce a report like:

```
-------------------- coverage: ... ---------------------
Name                    Stmts   Miss  Cover
--------------------------------------------
myproj/__init__             2      0    100%
myproj/myproj             257     13     94%
myproj/feature4286         94      7     92%
--------------------------------------------
TOTAL                     353     20     94%
```

1.3 Documentation

>   http://pytest-cov.rtfd.org/

1.4 Coverage Data File

The data file is erased at the beginning of testing to ensure clean data for each test run. If you need to combine the coverage of several test runs you can use the `--cov-append` option to append this coverage data to coverage data from previous test runs.

The data file is left at the end of testing so that it is possible to use normal coverage tools to examine it.

1.5 Limitations

For distributed testing the slaves must have the pytest-cov package installed. This is needed since the plugin must be registered through setuptools for pytest to start the plugin on the slave.

For subprocess measurement environment variables must make it from the main process to the subprocess. The python used by the subprocess must have pytest-cov installed. The subprocess must do normal site initialisation so that the environment variables can be detected and coverage started.

1.6 Acknowledgements

Whilst this plugin has been built fresh from the ground up it has been influenced by the work done on pytest-coverage (Ross Lawley, James Mills, Holger Krekel) and nose-cover (Jason Pellerin) which are other coverage plugins.

Ned Batchelder for coverage and its ability to combine the coverage results of parallel runs.

Holger Krekel for pytest with its distributed testing support.

Jason Pellerin for nose.

Michael Foord for unittest2.

No doubt others have contributed to these tools as well.

Configuration

This plugin provides a clean minimal set of command line options that are added to pytest. For further control of coverage use a coverage config file.

For example if tests are contained within the directory tree being measured the tests may be excluded if desired by using a .coveragerc file with the omit option set:

```
pytest --cov-config .coveragerc
        --cov=myproj
        myproj/tests/
```

Where the .coveragerc file contains file globs:

```
[run]
omit = tests/*
```

For full details refer to the coverage config file documentation.

Note that this plugin controls some options and setting the option in the config file will have no effect. These include specifying source to be measured (source option) and all data file handling (data_file and parallel options).

If you wish to always add pytest-cov with pytest, you can use `addopts` under `pytest` or `tool:pytest` section. For example:

```
[tool:pytest]
addopts = --cov=<project-name> --cov-report html
```

2.1 Caveats

A unfortunate consequence of coverage.py's history is that `.coveragerc` is a magic name: it's the default file but it also means "try to also lookup coverage configuration in `tox.ini` or `setup.cfg`".

In practical terms this means that if you have your coverage configuration in `tox.ini` or `setup.cfg` it is paramount that you also use `--cov-config=tox.ini` or `--cov-config=setup.cfg`.

You might not be affected but it's unlikely that you won't ever use `chdir` in a test.

Reporting

It is possible to generate any combination of the reports for a single test run.

The available reports are terminal (with or without missing line numbers shown), HTML, XML and annotated source code.

The terminal report without line numbers (default):

```
pytest --cov-report term --cov=myproj tests/

-------------------- coverage: platform linux2, python 2.6.4-final-0 ----------------
→----
Name                    Stmts   Miss  Cover
--------------------------------------------
myproj/__init__             2      0   100%
myproj/myproj             257     13    94%
myproj/feature4286         94      7    92%
--------------------------------------------
TOTAL                     353     20    94%
```

The terminal report with line numbers:

```
pytest --cov-report term-missing --cov=myproj tests/

-------------------- coverage: platform linux2, python 2.6.4-final-0 ----------------
→----
Name                    Stmts   Miss  Cover   Missing
-----------------------------------------------------
myproj/__init__             2      0   100%
myproj/myproj             257     13    94%   24-26, 99, 149, 233-236, 297-298, 369-370
myproj/feature4286         94      7    92%   183-188, 197
-----------------------------------------------------
TOTAL                     353     20    94%
```

The terminal report with skip covered:

```
pytest --cov-report term:skip-covered --cov=myproj tests/

-------------------- coverage: platform linux2, python 2.6.4-final-0 ------------------
↪----
Name                      Stmts    Miss   Cover
----------------------------------------------
myproj/myproj               257      13    94%
myproj/feature4286           94       7    92%
----------------------------------------------
TOTAL                       353      20    94%

1 files skipped due to complete coverage.
```

You can use `skip-covered` with `term-missing` as well. e.g. `--cov-report term-missing:skip-covered`

These three report options output to files without showing anything on the terminal:

```
pytest --cov-report html
       --cov-report xml
       --cov-report annotate
       --cov=myproj tests/
```

The output location for each of these reports can be specified. The output location for the XML report is a file. Where as the output location for the HTML and annotated source code reports are directories:

```
pytest --cov-report html:cov_html
       --cov-report xml:cov.xml
       --cov-report annotate:cov_annotate
       --cov=myproj tests/
```

The final report option can also suppress printing to the terminal:

```
pytest --cov-report= --cov=myproj tests/
```

This mode can be especially useful on continuous integration servers, where a coverage file is needed for subsequent processing, but no local report needs to be viewed. For example, tests run on Travis-CI could produce a .coverage file for use with Coveralls.

Debuggers and PyCharm

(or other IDEs)

When it comes to TDD one obviously would like to debug tests. Debuggers in Python use mostly the sys.settrace function to gain access to context. Coverage uses the same technique to get access to the lines executed. Coverage does not play well with other tracers simultaneously running. This manifests itself in behaviour that PyCharm might not hit a breakpoint no matter what the user does. Since it is common practice to have coverage configuration in the pytest.ini file and pytest does not support removeopts or similar the *–no-cov* flag can disable coverage completely.

At the reporting part a warning message will show on screen:

```
Coverage disabled via --no-cov switch!
```

CHAPTER 5

Distributed testing (xdist)

5.1 "load" mode

Distributed testing with dist mode set to load will report on the combined coverage of all slaves. The slaves may be spread out over any number of hosts and each slave may be located anywhere on the file system. Each slave will have its subprocesses measured.

Running distributed testing with dist mode set to load:

```
pytest --cov=myproj -n 2 tests/
```

Shows a terminal report:

```
-------------------- coverage: platform linux2, python 2.6.4-final-0 -----------------
↪----
Name                   Stmts   Miss  Cover
----------------------------------------
myproj/__init__            2      0   100%
myproj/myproj            257     13    94%
myproj/feature4286        94      7    92%
----------------------------------------
TOTAL                    353     20    94%
```

Again but spread over different hosts and different directories:

```
pytest --cov=myproj --dist load
        --tx ssh=memedough@host1//chdir=testenv1
        --tx ssh=memedough@host2//chdir=/tmp/testenv2//python=/tmp/env1/bin/python
        --rsyncdir myproj --rsyncdir tests --rsync examples
        tests/
```

Shows a terminal report:

```
-------------------- coverage: platform linux2, python 2.6.4-final-0 ------------------
↪----
Name                    Stmts   Miss  Cover
------------------------------------------
myproj/__init__             2      0   100%
myproj/myproj             257     13    94%
myproj/feature4286         94      7    92%
------------------------------------------
TOTAL                     353     20    94%
```

5.2 "each" mode

Distributed testing with dist mode set to each will report on the combined coverage of all slaves. Since each slave is running all tests this allows generating a combined coverage report for multiple environments.

Running distributed testing with dist mode set to each:

```
pytest --cov=myproj --dist each
        --tx popen//chdir=/tmp/testenv3//python=/usr/local/python27/bin/python
        --tx ssh=memedough@host2//chdir=/tmp/testenv4//python=/tmp/env2/bin/python
        --rsyncdir myproj --rsyncdir tests --rsync examples
        tests/
```

Shows a terminal report:

```
--------------------------------------- coverage --------------------------------------
↪----
                    platform linux2, python 2.6.5-final-0
                    platform linux2, python 2.7.0-final-0
Name                    Stmts   Miss  Cover
------------------------------------------
myproj/__init__             2      0   100%
myproj/myproj             257     13    94%
myproj/feature4286         94      7    92%
------------------------------------------
TOTAL                     353     20    94%
```

Multiprocessing support

Although pytest-cov supports multiprocessing there are few pitfalls that need to be explained.

6.1 Abusing `Process.terminate`

It appears that many people are using the `terminate` method and then get unreliable coverage results.

On Linux usually that means a SIGTERM gets sent to the process. Unfortunately Python don't have a default handler for SIGTERM so you need to install your own. Because `pytest-cov` doesn't want to second-guess (not yet, add your thoughts on the issue tracker if you disagree) it doesn't install a handler by default, but you can activate it by doing anything like:

```python
from pytest_cov.embed import cleanup_on_sigterm
cleanup_on_sigterm()

# alternatively you can do this

from pytest_cov.embed import cleanup

def my_handler(signum, frame):
    cleanup()
    # custom cleanup
signal.signal(signal.SIGTERM, my_handler)
```

On Windows there's no nice way to do cleanup (no signal handlers) so you're left to your own devices.

6.2 Ungraceful Pool shutdown

Another problem is when using the `Pool` object. If you run some work on a pool in a test you're not guaranteed to get all the coverage data unless you use the `join` method.

Eg:

```python
from multiprocessing import Pool


def f(x):
    return x*x

if __name__ == '__main__':
    with Pool(5) as p:
        print(p.map(f, [1, 2, 3]))

    p.join()  # <= THIS IS ESSENTIAL
```

6.3 Ungraceful Process shutdown

There's an identical issue when using the `Process` objects. Don't forget to use `.join()`:

```python
from multiprocessing import Process


def f(name):
    print('hello', name)

if __name__ == '__main__':
    p = Process(target=f, args=('bob',))
    p.start()

    p.join()  # <= THIS IS ESSENTIAL
```

Plugin coverage

Getting coverage on pytest plugins is a very particular situation. Because how pytest implements plugins (using setuptools entrypoints) it doesn't allow controlling the order in which the plugins load. See pytest/issues/935 for technical details.

The current way of dealing with this problem is using the append feature and manually starting `pytest-cov`'s engine, eg:

> COV_CORE_SOURCE=src COV_CORE_CONFIG=.coveragerc COV_CORE_DATAFILE=.coverage.eager
> pytest –cov=src –cov-append

Alternatively you can have this in `tox.ini` (if you're using Tox of course):

```
[testenv]
setenv =
    COV_CORE_SOURCE={toxinidir}/src
    COV_CORE_CONFIG={toxinidir}/.coveragerc
    COV_CORE_DATAFILE={toxinidir}/.coverage.eager
```

And in `pytest.ini` / `tox.ini` / `setup.cfg`:

```
[tool:pytest]
addopts = --cov --cov-append
```

CHAPTER 8

Markers and fixtures

There are some builtin markers and fixtures in `pytest-cov`.

8.1 Markers

8.1.1 `no_cover`

Eg:

```
@pytest.mark.no_cover
def test_foobar():
    # do some stuff that needs coverage disabled
```

> **Warning:** Caveat
>
> Note that subprocess coverage will also be disabled.

8.2 Fixtures

8.2.1 `no_cover`

Eg:

```
def test_foobar(no_cover):
    # same as the marker ...
```

8.2.2 `cov`

For reasons that no one can remember there is a `cov` fixture that provides access to the underlying Coverage instance. Some say this is a disguised foot-gun and should be removed, and some think mysteries make life more interesting and it should be left alone.

Changelog

9.1 2.6.1 (2019-01-07)

- Added support for Pytest 4.1. Contributed by Daniel Hahler and   in #253 and #230.

- Various test and docs fixes. Contributed by Daniel Hahler in #224 and #223.

- Fixed the "Module already imported" issue (#211). Contributed by Daniel Hahler in #228.

9.2 2.6.0 (2018-09-03)

- Dropped support for Python < 3.4, Pytest < 3.5 and Coverage < 4.4.

- Fixed some documentation formatting. Contributed by Jean Jordaan and Julian.

- Added an example with `addopts` in documentation. Contributed by Samuel Giffard in #195.

- Fixed `TypeError:  'NoneType' object is not iterable` in certain xdist configurations. Contributed by Jeremy Bowman in #213.

- Added a `no_cover` marker and fixture. Fixes #78.

- Fixed broken `no_cover` check when running doctests. Contributed by Terence Honles in #200.

- Fixed various issues with path normalization in reports (when combining coverage data from parallel mode). Fixes #130. Contributed by Ryan Hiebert & Ionel Cristian Mărieș in #178.

- Report generation failures don't raise exceptions anymore. A warning will be logged instead. Fixes #161.

- Fixed multiprocessing issue on Windows (empty env vars are not passed). Fixes #165.

9.3 2.5.1 (2017-05-11)

- Fixed xdist breakage (regression in `2.5.0`). Fixes #157.

- Allow setting custom `data_file` name in `.coveragerc`. Fixes #145. Contributed by Jannis Leidel & Ionel Cristian Mărieș in #156.

9.4 2.5.0 (2017-05-09)

- Always show a summary when `--cov-fail-under` is used. Contributed by Francis Niu in PR#141.
- Added `--cov-branch` option. Fixes #85.
- Improve exception handling in subprocess setup. Fixes #144.
- Fixed handling when `--cov` is used multiple times. Fixes #151.

9.5 2.4.0 (2016-10-10)

- Added a "disarm" option: `--no-cov`. It will disable coverage measurements. Contributed by Zoltan Kozma in PR#135.

  WARNING: Do not put this in your configuration files, it's meant to be an one-off for situations where you want to disable coverage from command line.

- Fixed broken exception handling on `.pth` file. See #136.

9.6 2.3.1 (2016-08-07)

- Fixed regression causing spurious errors when xdist was used. See #124.
- Fixed DeprecationWarning about incorrect *addoption* use. Contributed by Florian Bruhin in PR#127.
- Fixed deprecated use of funcarg fixture API. Contributed by Daniel Hahler in PR#125.

9.7 2.3.0 (2016-07-05)

- Add support for specifying output location for html, xml, and annotate report. Contributed by Patrick Lannigan in PR#113.
- Fix bug hiding test failure when cov-fail-under failed.
- For coverage >= 4.0, match the default behaviour of *coverage report* and error if coverage fails to find the source instead of just printing a warning. Contributed by David Szotten in PR#116.
- Fixed bug occurred when bare `--cov` parameter was used with xdist. Contributed by Michael Elovskikh in PR#120.
- Add support for `skip_covered` and added `--cov-report=term-skip-covered` command line options. Contributed by Saurabh Kumar in PR#115.

9.8 2.2.1 (2016-01-30)

- Fixed incorrect merging of coverage data when xdist was used and coverage was `>= 4.0`.

9.9 2.2.0 (2015-10-04)

- Added support for changing working directory in tests. Previously changing working directory would disable coverage measurements in suprocesses.

- Fixed broken handling for `--cov-report=annotate`.

9.10 2.1.0 (2015-08-23)

- Added support for *coverage 4.0b2*.

- Added the `--cov-append` command line options. Contributed by Christian Ledermann in PR#80.

9.11 2.0.0 (2015-07-28)

- Added `--cov-fail-under`, akin to the new `fail_under` option in *coverage-4.0* (automatically activated if there's a `[report] fail_under = ...` in `.coveragerc`).

- Changed `--cov-report=term` to automatically upgrade to `--cov-report=term-missing` if there's `[run] show_missing = True` in `.coveragerc`.

- Changed `--cov` so it can be used with no path argument (in wich case the source settings from `.coveragerc` will be used instead).

- Fixed *.pth* installation to work in all cases (install, easy_install, wheels, develop etc).

- Fixed *.pth* uninstallation to work for wheel installs.

- Support for coverage 4.0.

- Data file suffixing changed to use coverage's `data_suffix=True` option (instead of the custom suffixing).

- Avoid warning about missing coverage data (just like `coverage.control.process_startup`).

- Fixed a race condition when running with xdist (all the workers tried to combine the files). It's possible that this issue is not present in *pytest-cov 1.8.X*.

9.12 1.8.2 (2014-11-06)

- N/A

Authors

- Marc Schlaich - http://www.schlamar.org
- Rick van Hattem - http://wol.ph
- Buck Evan - https://github.com/bukzor
- Eric Larson - http://larsoner.com
- Marc Abramowitz - http://marc-abramowitz.com
- Thomas Kluyver - https://github.com/takluyver
- Guillaume Ayoub - http://www.yabz.fr
- Federico Ceratto - http://firelet.net
- Josh Kalderimis - http://blog.cookiestack.com
- Ionel Cristian Mărieș - https://blog.ionelmc.ro
- Christian Ledermann - https://github.com/cleder
- Alec Nikolas Reiter - https://github.com/justanr
- Patrick Lannigan - https://github.com/plannigan
- David Szotten - https://github.com/davidszotten
- Michael Elovskikh - https://github.com/wronglink
- Saurabh Kumar - https://github.com/theskumar
- Michael Elovskikh - https://github.com/wronglink
- Daniel Hahler - https://daniel.hahler.de
- Florian Bruhin - http://www.the-compiler.org
- Zoltan Kozma - https://github.com/kozmaz87
- Francis Niu - https://flniu.github.io
- Jannis Leidel - https://github.com/jezdez

- Ryan Hiebert - http://ryanhiebert.com/
- Terence Honles - https://github.com/terencehonles
- Jeremy Bowman - https://github.com/jmbowman
- Samuel Giffard - https://github.com/Mulugruntz
- - https://github.com/MarSoft

Releasing

The process for releasing should follow these steps:

1. Test that docs build and render properly by running `tox -e docs,spell`.

    If there are bogus spelling issues add the words in `spelling_wordlist.txt`.

2. Update `CHANGELOG.rst` and `AUTHORS.rst` to be up to date.

3. Bump the version by running `bumpversion [ major | minor | patch ]`. This will automatically add a tag.

    Alternatively, you can manually edit the files and run `git tag v1.2.3` yourself.

4. Push changes and tags with:

    ```
    git push
    git push --tags
    ```

5. Wait for AppVeyor and Travis to give the green builds.

6. Check that the docs on ReadTheDocs are built.

7. Make sure you have a clean checkout, run `git status` to verify.

8. Manually clean temporary files (that are ignored and won't show up in `git status`):

    ```
    rm -rf dist build src/*.egg-info
    ```

    These files need to be removed to force distutils/setuptools to rebuild everything and recreate the egg-info metadata.

9. Build the dists:

    ```
    python3.4 setup.py clean --all sdist bdist_wheel
    ```

10. Verify that the resulting archives (found in `dist/`) are good.

11. Upload the sdist and wheel with twine:

```
twine upload dist/*
```

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

12.1 Bug reports

When 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.

12.2 Documentation improvements

pytest-cov could always use more documentation, whether as part of the official pytest-cov docs, in docstrings, or even on the web in blog posts, articles, and such.

12.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/pytest-dev/pytest-cov/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 code contributions are welcome :)

12.4 Development

To set up *pytest-cov* for local development:

1. Fork pytest-cov (look for the "Fork" button).

2. Clone your fork locally:

```
git clone git@github.com:your_name_here/pytest-cov.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. 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
```

6. Submit a pull request through the GitHub website.

12.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run tox)[1].

2. Update documentation when there's new API, functionality etc.

3. Add a note to CHANGELOG.rst about the changes.

4. Add yourself to AUTHORS.rst.

12.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel* (you need to pip install detox):

```
detox
```

---

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.

It will be slower though ...

CHAPTER 13

Indices and tables

- genindex
- modindex
- search