pytest-cov

Release 5.0.0

pytest-cov contributors

Mar 24, 2024

CONTENTS

Contents:

CHAPTER

ONE

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

https://pytest-cov.readthedocs.io/en/latest/

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 workers 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 worker.

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.

CHAPTER

TWO

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: Important Note

This plugin overrides the `parallel` option of coverage. Unless you also run coverage without pytest-cov it's pointless to set those options in your `.coveragerc`.

If you use the `--cov=something` option (with a value) then coverage's `source` option will also get overridden. If you have multiple sources it might be easier to set those in `.coveragerc` and always use `--cov` (without a value) instead of having a long command line with `--cov=pkg1 --cov=pkg2 --cov=pkg3 ...`.

If you use the `--cov-branch` option then coverage's `branch` option will also get overridden.

If you wish to always add pytest-cov with pytest, you can use `addopts` under the `pytest` or `tool:pytest` section of your `setup.cfg`, or the `tool.pytest.ini_options` section of your `pyproject.toml` file.

For example, in `setup.cfg`:

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

Or for `pyproject.toml`:

```
[tool.pytest.ini_options]
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.

2.2 Reference

The complete list of command line options is:

--cov=PATH	Measure coverage for filesystem path. (multi-allowed)
--cov-report=type	Type of report to generate: term, term-missing, annotate, html, xml, json, lcov (multi-allowed). term, term- missing may be followed by ":skip-covered". annotate, html, xml, json and lcov may be followed by ":DEST" where DEST specifies the output location. Use –cov-report= to not generate any output.
--cov-config=path	Config file for coverage. Default: .coveragerc
--no-cov-on-fail	Do not report coverage if test run fails. Default: False
--no-cov	Disable coverage report completely (useful for debuggers). Default: False
--cov-reset	Reset cov sources accumulated in options so far. Mostly useful for scripts and configuration files.
--cov-fail-under=MIN	Fail if the total coverage is less than MIN.
--cov-append	Do not delete coverage but append to current. Default: False
--cov-branch	Enable branch coverage.
--cov-context	Choose the method for setting the dynamic context.

CHAPTER

THREE

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, JSON, LCOV 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%
```

(continues on next page)

(continued from previous page)

```
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 four report options output to files without showing anything on the terminal:

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

The output location for each of these reports can be specified. The output location for the XML, JSON and LCOV 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 json:cov.json
       --cov-report lcov:cov.info
       --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 GitHub Actions could produce a .coverage file for use with Coveralls.

CHAPTER

FOUR

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, or encountering an error like this:

```
PYDEV DEBUGGER WARNING:
sys.settrace() should not be used when the debugger is being used.
This may cause the debugger to stop working correctly.
```

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

FIVE

DISTRIBUTED TESTING (XDIST)

5.1 “load” mode

Distributed testing with dist mode set to “load” will report on the combined coverage of all workers. The workers may be spread out over any number of hosts and each worker may be located anywhere on the file system. Each worker 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 workers. Since each worker 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%
```

CHAPTER

SIX

SUBPROCESS SUPPORT

Normally coverage writes the data via a pretty standard atexit handler. However, if the subprocess doesn't exit on its own then the atexit handler might not run. Why that happens is best left to the adventurous to discover by waddling through the Python bug tracker.

pytest-cov supports subprocesses and multiprocessing, and works around these atexit limitations. However, there are a few pitfalls that need to be explained.

6.1 If you use `multiprocessing.Pool`

pytest-cov automatically registers a multiprocessing finalizer. The finalizer will only run reliably if the pool is closed. Closing the pool basically signals the workers that there will be no more work, and they will eventually exit. Thus one also needs to call *join* on the pool.

If you use `multiprocessing.Pool.terminate` or the context manager API (`__exit__` will just call `terminate`) then the workers can get SIGTERM and then the finalizers won't run or complete in time. Thus you need to make sure your `multiprocessing.Pool` gets a nice and clean exit:

```
from multiprocessing import Pool

def f(x):
    return x*x

if __name__ == '__main__':
    p = Pool(5)
    try:
        print(p.map(f, [1, 2, 3]))
    finally:
        p.close()  # Marks the pool as closed.
        p.join()   # Waits for workers to exit.
```

If you must use the context manager API (e.g.: the pool is managed in third party code you can't change) then you can register a cleaning SIGTERM handler like so:

Warning: **This technique cannot be used on Python 3.8** (registering signal handlers will cause deadlocks in the pool, see: https://bugs.python.org/issue38227).

```
from multiprocessing import Pool
```

(continues on next page)

(continued from previous page)

```
def f(x):
    return x*x

if __name__ == '__main__':
    try:
        from pytest_cov.embed import cleanup_on_sigterm
    except ImportError:
        pass
    else:
        cleanup_on_sigterm()

    with Pool(5) as p:
        print(p.map(f, [1, 2, 3]))
```

6.2 If you use `multiprocessing.Process`

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

```
from multiprocessing import Process

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

if __name__ == '__main__':
    try:
        from pytest_cov.embed import cleanup_on_sigterm
    except ImportError:
        pass
    else:
        cleanup_on_sigterm()

    p = Process(target=f, args=('bob',))
    try:
        p.start()
    finally:
        p.join()  # necessary so that the Process exists before the test suite exits␣
↪(thus coverage is collected)
```

6.3 If you got custom signal handling

pytest-cov 2.6 has a rudimentary `pytest_cov.embed.cleanup_on_sigterm` you can use to register a SIGTERM handler that flushes the coverage data.

pytest-cov 2.7 adds a `pytest_cov.embed.cleanup_on_signal` function and changes the implementation to be more robust: the handler will call the previous handler (if you had previously registered any), and is re-entrant (will defer extra signals if delivered while the handler runs).

For example, if you reload on SIGHUP you should have something like this:

```
import os
import signal

def restart_service(frame, signum):
    os.exec( ... )  # or whatever your custom signal would do
signal.signal(signal.SIGHUP, restart_service)

try:
    from pytest_cov.embed import cleanup_on_signal
except ImportError:
    pass
else:
    cleanup_on_signal(signal.SIGHUP)
```

Note that both `cleanup_on_signal` and `cleanup_on_sigterm` will run the previous signal handler.

Alternatively you can do this:

```
import os
import signal

try:
    from pytest_cov.embed import cleanup
except ImportError:
    cleanup = None

def restart_service(frame, signum):
    if cleanup is not None:
        cleanup()

    os.exec( ... )  # or whatever your custom signal would do
signal.signal(signal.SIGHUP, restart_service)
```

6.4 If you use Windows

On Windows you can register a handler for SIGTERM but it doesn't actually work. It will work if you *os.kill(os.getpid(), signal.SIGTERM)* (send SIGTERM to the current process) but for most intents and purposes that's completely useless.

Consequently this means that if you use multiprocessing you got no choice but to use the close/join pattern as described above. Using the context manager API or *terminate* won't work as it relies on SIGTERM.

However you can have a working handler for SIGBREAK (with some caveats):

```
import os
import signal

def shutdown(frame, signum):
    # your app's shutdown or whatever
signal.signal(signal.SIGBREAK, shutdown)

try:
    from pytest_cov.embed import cleanup_on_signal
```

(continues on next page)

(continued from previous page)

```
except ImportError:
    pass
else:
    cleanup_on_signal(signal.SIGBREAK)
```

The caveats being roughly:

- you need to deliver `signal.CTRL_BREAK_EVENT`
- it gets delivered to the whole process group, and that can have unforeseen consequences

CHAPTER

SEVEN

CONTEXTS

Coverage.py 5.0 can record separate coverage data for different contexts during one run of a test suite. Pytest-cov can use this feature to record coverage data for each test individually, with the `--cov-context=test` option.

The context name recorded in the coverage.py database is the pytest test id, and the phase of execution, one of "setup", "run", or "teardown". These two are separated with a pipe symbol. You might see contexts like:

```
test_functions.py::test_addition|run
test_fancy.py::test_parametrized[1-101]|setup
test_oldschool.py::RegressionTests::test_error|run
```

Note that parameterized tests include the values of the parameters in the test id, and each set of parameter values is recorded as a separate test.

To view contexts when using `--cov-report=html`, add this to your `.coveragerc`:

```
[html]
show_contexts = True
```

The HTML report will include an annotation on each covered line, indicating the number of contexts that executed the line. Clicking the annotation displays a list of the contexts.

CHAPTER

EIGHT

TOX

When using tox you can have ultra-compact configuration - you can have all of it in `tox.ini`:

```
[tox]
envlist = ...

[tool:pytest]
...

[coverage:paths]
...

[coverage:run]
...

[coverage:report]
..

[testenv]
commands = ...
```

An usual problem users have is that pytest-cov will erase the previous coverage data by default, thus if you run tox with multiple environments you'll get incomplete coverage at the end.

To prevent this problem you need to use `--cov-append`. It's still recommended to clean the previous coverage data to have consistent output. A `tox.ini` like this should be enough for sequential runs:

```
[tox]
envlist = clean,py27,py36,...

[testenv]
commands = pytest --cov --cov-append --cov-report=term-missing ...
deps =
    pytest
    pytest-cov

[testenv:clean]
deps = coverage
skip_install = true
commands = coverage erase
```

For parallel runs we need to set some dependencies and have an extra report env like so:

```
[tox]
envlist = clean,py27,py36,report

[testenv]
commands = pytest --cov --cov-append --cov-report=term-missing
deps =
    pytest
    pytest-cov
depends =
    {py27,py36}: clean
    report: py27,py36

[testenv:report]
deps = coverage
skip_install = true
commands =
    coverage report
    coverage html

[testenv:clean]
deps = coverage
skip_install = true
commands = coverage erase
```

Depending on your project layout you might need extra configuration, see the working examples at https://github.com/pytest-dev/pytest-cov/tree/master/examples for two common layouts.

CHAPTER

NINE

PLUGIN COVERAGE

Getting coverage on pytest plugins is a very particular situation. Because of 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=
    COV_CORE_CONFIG={toxinidir}/.coveragerc
    COV_CORE_DATAFILE={toxinidir}/.coverage
```

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

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

CHAPTER

TEN

MARKERS AND FIXTURES

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

10.1 Markers

10.1.1 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.

10.2 Fixtures

10.2.1 no_cover

Eg:

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

10.2.2 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.

CHAPTER

ELEVEN

CHANGELOG

11.1 5.0.0 (2024-03-24)

- Removed support for xdist rsync (now deprecated). Contributed by Matthias Reichenbach in #623.
- Switched docs theme to Furo.
- Various legacy Python cleanup and CI improvements. Contributed by Christian Clauss and Hugo van Kemenade in #630, #631, #632 and #633.
- Added a `pyproject.toml` example in the docs. Contributed by Dawn James in #626.
- Modernized project's pre-commit hooks to use ruff. Initial POC contributed by Christian Clauss in #584.

11.2 4.1.0 (2023-05-24)

- Updated CI with new Pythons and dependencies.
- Removed rsyncdir support. This makes pytest-cov compatible with xdist 3.0. Contributed by Sorin Sbarnea in #558.
- Optimized summary generation to not be performed if no reporting is active (for example, when `--cov-report=''` is used without `--cov-fail-under`). Contributed by Jonathan Stewmon in #589.
- Added support for JSON reporting. Contributed by Matthew Gamble in #582.
- Refactored code to use f-strings. Contributed by Mark Mayo in #572.
- Fixed a skip in the test suite for some old xdist. Contributed by a bunch of people in #565.

11.3 4.0.0 (2022-09-28)

Note that this release drops support for multiprocessing.

- *–cov-fail-under* no longer causes *pytest –collect-only* to fail Contributed by Zac Hatfield-Dodds in #511.
- Dropped support for multiprocessing (mostly because issue 82408). This feature was mostly working but very broken in certain scenarios and made the test suite very flaky and slow.

  There is builtin multiprocessing support in coverage and you can migrate to that. All you need is this in your `.coveragerc`:

```
[run]
concurrency = multiprocessing
parallel = true
sigterm = true
```

- Fixed deprecation in `setup.py` by trying to import setuptools before distutils. Contributed by Ben Greiner in #545.
- Removed undesirable new lines that were displayed while reporting was disabled. Contributed by Delgan in #540.
- Documentation fixes. Contributed by Andre Brisco in #543 and Colin O'Dell in #525.
- Added support for LCOV output format via *–cov-report=lcov*. Only works with coverage 6.3+. Contributed by Christian Fetzer in #536.
- Modernized pytest hook implementation. Contributed by Bruno Oliveira in #549 and Ronny Pfannschmidt in #550.

11.4 3.0.0 (2021-10-04)

Note that this release drops support for Python 2.7 and Python 3.5.

- Added support for Python 3.10 and updated various test dependencies. Contributed by Hugo van Kemenade in #500.
- Switched from Travis CI to GitHub Actions. Contributed by Hugo van Kemenade in #494 and #495.
- Add a `--cov-reset` CLI option. Contributed by Danilo Šegan in #459.
- Improved validation of `--cov-fail-under` CLI option. Contributed by … Ronny Pfannschmidt's desire for skark in #480.
- Dropped Python 2.7 support. Contributed by Thomas Grainger in #488.
- Updated trove classifiers. Contributed by Michał Bielawski in #481.
- Reverted change for *toml* requirement. Contributed by Thomas Grainger in #477.

11.5 2.12.1 (2021-06-01)

- Changed the *toml* requirement to be always be directly required (instead of being required through a coverage extra). This fixes issues with pip-compile (pip-tools#1300). Contributed by Sorin Sbarnea in #472.
- Documented `show_contexts`. Contributed by Brian Rutledge in #473.

11.6 2.12.0 (2021-05-14)

- Added coverage's *toml* extra to install requirements in setup.py. Contributed by Christian Riedel in #410.
- Fixed `pytest_cov.__version__` to have the right value (string with version instead of a string including `__version__ =`).
- Fixed license classifier in `setup.py`. Contributed by Chris Sreesangkom in #467.
- Fixed *commits since* badge. Contributed by Terence Honles in #470.

11.7 2.11.1 (2021-01-20)

- Fixed support for newer setuptools (v42+). Contributed by Michał Górny in #451.

11.8 2.11.0 (2021-01-18)

- Bumped minimum coverage requirement to 5.2.1. This prevents reporting issues. Contributed by Mateus Berardo de Souza Terra in #433.
- Improved sample projects (from the examples directory) to support running *tox -e pyXY*. Now the example configures a suffixed coverage data file, and that makes the cleanup environment unnecessary. Contributed by Ganden Schaffner in #435.
- Removed the empty *console_scripts* entrypoint that confused some Gentoo build script. I didn't ask why it was so broken cause I didn't want to ruin my day. Contributed by Michał Górny in #434.
- Fixed the missing coverage context when using subprocesses. Contributed by Bernát Gábor in #443.
- Updated the config section in the docs. Contributed by Pamela McA'Nulty in #429.
- Migrated CI to travis-ci.com (from .org).

11.9 2.10.1 (2020-08-14)

- Support for `pytest-xdist` 2.0, which breaks compatibility with `pytest-xdist` before 1.22.3 (from 2017). Contributed by Zac Hatfield-Dodds in #412.
- Fixed the `LocalPath has no attribute startswith` failure that occurred when using the `pytester` plugin in inline mode.

11.10 2.10.0 (2020-06-12)

- Improved the `--no-cov` warning. Now it's only shown if `--no-cov` is present before `--cov`.
- Removed legacy pytest support. Changed `setup.py` so that `pytest>=4.6` is required.

11.11 2.9.0 (2020-05-22)

- Fixed `RemovedInPytest4Warning` when using Pytest 3.10. Contributed by Michael Manganiello in #354.
- Made pytest startup faster when plugin not active by lazy-importing. Contributed by Anders Hovmöller in #339.
- Various CI improvements. Contributed by Daniel Hahler in #363 and #364.
- Various Python support updates (drop EOL 3.4, test against 3.8 final). Contributed by Hugo van Kemenade in #336 and #367.
- Changed `--cov-append` to always enable `data_suffix` (a coverage setting). Contributed by Harm Geerts in #387.
- Changed `--cov-append` to handle loading previous data better (fixes various path aliasing issues).
- Various other testing improvements, github issue templates, example updates.
- Fixed internal failures that are caused by tests that change the current working directory by ensuring a consistent working directory when coverage is called. See #306 and coveragepy#881

11.12 2.8.1 (2019-10-05)

- Fixed #348 - regression when only certain reports (html or xml) are used then `--cov-fail-under` always fails.

11.13 2.8.0 (2019-10-04)

- Fixed `RecursionError` that can occur when using cleanup_on_signal or cleanup_on_sigterm. See: #294. The 2.7.x releases of pytest-cov should be considered broken regarding aforementioned cleanup API.
- Added compatibility with future xdist release that deprecates some internals (match pytest-xdist master/worker terminology). Contributed by Thomas Grainger in #321
- Fixed breakage that occurs when multiple reporting options are used. Contributed by Thomas Grainger in #338.
- Changed internals to use a stub instead of `os.devnull`. Contributed by Thomas Grainger in #332.
- Added support for Coverage 5.0. Contributed by Ned Batchelder in #319.
- Added support for float values in `--cov-fail-under`. Contributed by Martín Gaitán in #311.
- Various documentation fixes. Contributed by Juanjo Bazán, Andrew Murray and Albert Tugushev in #298, #299 and #307.
- Various testing improvements. Contributed by Ned Batchelder, Daniel Hahler, Ionel Cristian Mărieș and Hugo van Kemenade in #313, #314, #315, #316, #325, #326, #334 and #335.
- Added the `--cov-context` CLI options that enables coverage contexts. Only works with coverage 5.0+. Contributed by Ned Batchelder in #345.

11.14 2.7.1 (2019-05-03)

- Fixed source distribution manifest so that garbage ain't included in the tarball.

11.15 2.7.0 (2019-05-03)

- Fixed `AttributeError: 'NoneType' object has no attribute 'configure_node'` error when `--no-cov` is used. Contributed by Alexander Shadchin in #263.
- Various testing and CI improvements. Contributed by Daniel Hahler in #255, #266, #272, #271 and #269.
- Improved `pytest_cov.embed.cleanup_on_sigterm` to be reentrant (signal deliveries while signal handling is running won't break stuff).
- Added `pytest_cov.embed.cleanup_on_signal` for customized cleanup.
- Improved cleanup code and fixed various issues with leftover data files. All contributed in #265 or #262.
- Improved examples. Now there are two examples for the common project layouts, complete with working coverage configuration. The examples have CI testing. Contributed in #267.
- Improved help text for CLI options.

11.16 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.

11.17 2.6.0 (2018-09-03)

- Dropped support for Python 3 < 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.

11.18 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.

11.19 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.

11.20 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.

11.21 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.

11.22 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.

11.23 2.2.1 (2016-01-30)

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

11.24 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`.

11.25 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.

11.26 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 which 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*.

11.27 1.8.2 (2014-11-06)

- N/A

CHAPTER

TWELVE

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

- Alexander Shadchin - https://github.com/shadchin
- Thomas Grainger - https://graingert.co.uk
- Juanjo Bazán - https://github.com/xuanxu
- Andrew Murray - https://github.com/radarhere
- Ned Batchelder - https://nedbatchelder.com/
- Albert Tugushev - https://github.com/atugushev
- Martín Gaitán - https://github.com/mgaitan
- Hugo van Kemenade - https://github.com/hugovk
- Michael Manganiello - https://github.com/adamantike
- Anders Hovmöller - https://github.com/boxed
- Zac Hatfield-Dodds - https://zhd.dev
- Mateus Berardo de Souza Terra - https://github.com/MatTerra
- Ganden Schaffner - https://github.com/gschaffner
- Michał Górny - https://github.com/mgorny
- Bernát Gábor - https://github.com/gaborbernat
- Pamela McA'Nulty - https://github.com/PamelaM
- Christian Riedel - https://github.com/Cielquan
- Chris Sreesangkom - https://github.com/csreesan
- Sorin Sbarnea - https://github.com/ssbarnea
- Brian Rutledge - https://github.com/bhrutledge
- Danilo Šegan - https://github.com/dsegan
- Michał Bielawski - https://github.com/D3X
- Zac Hatfield-Dodds - https://github.com/Zac-HD
- Ben Greiner - https://github.com/bnavigator
- Delgan - https://github.com/Delgan
- Andre Brisco - https://github.com/abrisco
- Colin O'Dell - https://github.com/colinodell
- Ronny Pfannschmidt - https://github.com/RonnyPfannschmidt
- Christian Fetzer - https://github.com/fetzerch
- Jonathan Stewmon - https://github.com/jstewmon
- Matthew Gamble - https://github.com/mwgamble
- Christian Clauss - https://github.com/cclauss
- Dawn James - https://github.com/dawngerpony

CHAPTER

THIRTEEN

RELEASING

The process for releasing should follow these steps:

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

   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.

4. Push changes and tags with:

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

5. Wait GitHub Actions 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 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/*
    ```

CHAPTER

FOURTEEN

CONTRIBUTING

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

14.1 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.

14.2 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.

14.3 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 :)

14.4 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:YOURGITHUBNAME/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 and docs builder with 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.

14.4.1 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`).
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`.

14.4.2 Tips

To run a subset of tests:

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

To run all the test environments in *parallel*:

```
tox -p auto
```

CHAPTER

FIFTEEN

INDICES AND TABLES

- genindex
- modindex
- search