Package ‘testthis’

May 8, 2026

Type Package

Title Utils and 'RStudio' Addins to Make Testing Even More Fun

Version 1.1.1

Maintainer Stefan Fleck <stefan.b.fleck@gmail.com>

Description Utility functions and 'RStudio' addins for writing, running and organizing automated tests. Integrates tightly with the packages 'testthat', 'devtools' and 'usethis'. Hotkeys can be assigned to the 'RStudio' addins for running tests in a single file or to switch between a source file and the associated test file. In addition, testthis provides function to manage and run tests in subdirectories of the test/testthat directory.

License MIT + file LICENSE

URL https://s-fleck.github.io/testthis

BugReports https://github.com/s-fleck/testthis/issues

Imports assertthat, devtools, fs, magrittr, pkgload, rprojroot, stats, stringi, testthat, tools, usethis (>= 0.1.0), utils

Suggests covr, crayon, knitr, rmarkdown, roxygen2 (>= 5.0.0), rstudioapi

Encoding UTF-8

RoxygenNote 7.1.0

NeedsCompilation no

Author Stefan Fleck [aut, cre] (ORCID: <https://orcid.org/0000-0003-3344-9851>)

Repository CRAN

Date/Publication 2020-04-12 06:30:02 UTC

Contents

get_test_coverage 2
open_testfile 2

testthis 3
test_coverage 4
test_index 5
test_skeleton 5
test_subdir 6
test_this 7
use_testdata 8
use_tester 9
use_test_subdir 10

Index **12**

`get_test_coverage` *Get Test Coverage*

Description

Deprecated in favor of `test_coverage()`

Usage

```
get_test_coverage(from_tags = TRUE, from_desc = TRUE)
```

Arguments

`from_tags`, `from_desc`
: see `test_coverage()`

`open_testfile` *Open associated test file*

Description

If the currently open file in the RStudio editor is called '`myfun.R`' this opens '`tests/testthat/test_myfun.R`' in a new tab. This function can also be used to jump back and forth between a script and the associated test file. You can modify this behaviour by putting the comment `#* @testfile anotherfile` anywhere in '`myfun.R`'.

Usage

```
open_testfile()
```

testthis *Testthis-package*

Description

RStudio addins for several common testing-related tasks during package development, such as switching between a source file and an associated test file, or running unit tests in a single test file (hotkeyable in RStudio!). testthis also provides utility function to manage tests in subdirectories of the test/testthis directory.

Details

For details please refer to the README

Setting the project path

Testthis uses `usethis::proj_get()` to detect project root of the current working directory. You can override the project root with `usethis::proj_set()`.

Package options

Package options can be set with `options()`. You can add `options()` to your '`.Rprofile`' to make them permanent across sessions.

`testthis.sep` Default separator to use when creating test files with `test_skeleton()`. Defaults to _, must be either _ or -; i.e whether you want your files to be named `test_foofunction.R` or `test-foofunction.R`

`testthis.integration_tests_path`, `testthis.acceptance_tests_path`, `testthis.manual_tests_path` Default paths used by the functions `testthis::use_integration_tests()`, `testthis::test_integration()`, etc...

Testthis Tags

test_this tags are special comments that modify the behaviour of the functions supplied by this package. They are of the form `#* @tag <value>`. Please not that only some test_this tags really require a `<value>`.

Valid tags for script files in the /R/ dir (`pkg/R/*.R`)

- `@testfile <filename>`: manually specify associated test file. Should usually start with `test_`. This is used by `test_this()`, `lest_this()` and `open_testfile()`.

Valid tags or test files (`pkg/tests/testthat/test_*.R`)

- `@testing <functionname>`: mark `functionname` as tested. Should usually go next the associated `test_that()` call. This is used by `test_coverage()`.

Author(s)

Maintainer: Stefan Fleck `<stefan.b.fleck@gmail.com>` (ORCID)

See Also

usethis::edit_r_profile()

---

test_coverage *Test coverage of package*

Description

This determines the test coverage of the target package based on the desc argument of test_that() calls. If you require a more comprehensive analysis of test coverage, try the package **covr** instead.

Usage

```
test_coverage(from_tags = TRUE, from_desc = TRUE)
```

Arguments

from_tags	logical scalar. Checks the files if your test directory for testthis tags. Specifically, if you have the comment #* @testing myfunction in any of your test files, myfunction will be marked as tested.
from_desc	logical scalar. Checks the desc argument test_that(...) of the tests in your test directory for functions names. E.g. if you have a test file that contains test_that("myfunction works", {...}), myfunction will be marked as tested.

Details

test_coverage looks in .covrignore for functions that should be ignored for coverage analysis (see usethis::use_covr_ignore())

Value

A Test_coverage object. This is a data.frame containing the following columns:

- fun: Name of the function
- exp: Is function is exported?
- s3: Is function an S3 method?
- tested: Do unit tests exist for function?
- ignore: Is function listed in 'tests/testthat/_testignore'?

Examples

```
## Not run:
x <- test_coverage()
as.data.frame(x)

## End(Not run)
```

test_index — *Generate an index of all test_that calls*

Description

Generates an index the `desc` arguments of all `test_that()` calls in the `tests/testthat` directory of a package.

Usage

```
test_index(
  markers = interactive() && requireNamespace("rstudioapi", quietly = TRUE)
)

## S3 method for class 'test_index'
print(x, ...)
```

Arguments

`markers`	`logical` scalar. If `TRUE`, new markers are created in the RStudio markers pane. If `FALSE`, the index is printed to the console instead.
`x`	a `data.frame` of subclass `"test_index"`
`...`	currently ignored

Value

A `test_index` `data.frame` (invisibly if `markers == TRUE`)

test_skeleton — *Create a test skeleton file for the currently open .R file*

Description

If the file currently open in the RStudio editor is called `my_function.R`, this creates the file '`/tests/testthat/test_my_fur` and fills it with a basic test skeleton.

Usage

```
test_skeleton(fname = NULL, open = TRUE, sep = options("testthis.sep"))
```

Arguments

`fname`	`character` scalar. Target R script file to open. If empty the file currently open in the editor will be used.
`open`	`logical` scalar. Should the test file be opened after it is created?
`sep`	`character` scalar. Separator between '`test`' and '`fname`' when constructing the test file name. Should either be `"_"` or `"-"` for compatibility with testthat.

Value

NULL (invisibly)

Side effects

Creates an R script file in the file system.

See Also

usethis::use_test()

---

test_subdir *Run tests in subdirectories*

Description

This is a simple wrapper for devtools::test(), but rather than running the tests in 'inst/tests/' or 'tests/testthat/', it runs the tests in a subdirectory of that folder. For creating such subdirectories, please also see use_test_subdir().

Usage

```
test_subdir(subdir, ...)

test_integration(...)

test_acceptance(...)

test_manual(...)

test_all(...)
```

Arguments

subdir	character scalar. subdir of inst/tests/ or tests/testthat that contains the tests
...	passed on to devtools::test()

Value

A testthat::testthat_results object (invisibly)

Test subdirectory presets

Three preset test subdirs are defined at the moment:

`test_integration()` Integration tests, also called component tests. Put tests here that test if several functions / parts of your program work together as expected. You can create the relevant subdir '`testthat/integration_tests/`' with `use_integration_tests()`.

`test_acceptance()` Acceptance tests. This is the highest levels of tests. Put tests here that verifies if your package fulfills the goals/requirements you set out to achieve with your package were met. You can create the relevant subdir '`testthat/acceptance_tests/`' with `use_acceptance_tests()`.

`test_manual()` Manual tests. Put tests here that produce output that has to be manually verified, such as: console output, pdf files, plots. It is recommended you collect the output files of such tests in '`tests/testthat/testout`'. You can create the relevant subdir with '`testthat/manual_tests/`' with `use_manual_tests()`.

You can modify the default paths for manual, acceptance and integration tests by setting the respective `options()`, but it is recommended to create your own test subdirs instead.

See Also

`use_test_subdir()`

---

test_this *Test this file*

Description

Runs testthat tests in a single `.R` file. If the file currently open in the RStudio editor is called `my_function.R`, `test_this()` calls `testthat::test_file()` on '`tests/testthat/test_my_function.R`'. If the filename of the currently open file with starts with `test_` it will call `testthat::test_file()` on the current file.

Usage

```
test_this(...)
```

Arguments

`...`	passed on to `testthat::test_file()`

Details

This is useful in cases where you don't want to run all tests in a package via `devtools::test()` (CTRL+SHIFT+T in RStudio).

Value

`NULL` (invisibly)

use_testdata *Create* testdata *folder.*

Description

Save R objects to separate files ‘tests/testthat/testdata’ in the .rds format.

Usage

```
use_testdata(
  ...,
  subdir = NULL,
  overwrite = FALSE,
  ignore = FALSE,
  compress = TRUE,
  refhook = NULL,
  version = NULL
)

use_testdata_raw(name = NULL)

has_testdata()

read_testdata(infile, subdir = NULL)

find_testdata(..., path = ".", must_exist = FALSE)
```

Arguments

...	R objects to save to the ‘testdata’ dir. If empty, an empty directory is created.
subdir	character scalar. Subdirectory of ‘testdata’ to save to / read from.
overwrite	Logical scalar. Should existing files be overwritten?
ignore	Should the newly created file be added to .Rbuildignore?
compress	a logical specifying whether saving to a named file is to use "gzip" compression, or one of "gzip", "bzip2" or "xz" to indicate the type of compression to be used. Ignored if file is a connection.
refhook	a hook function for handling reference objects.
version	the workspace format version to use. NULL specifies the current default version (3). The only other supported value is 2, the default from R 1.4.0 to R 3.5.0.
name	a character scalar or NULL. Name of the dataset for which to create a script file in ‘testthat/testdata-raw’. If NULL an empty ‘testthat/testdata-raw’ directory is created (if it does not exist already).
infile	rds file to read (must end in .rds, otherwise .rds ending is automatically added)
path	Path of the directory to create, relative to the project.
must_exist	logical scalar. Assert that path specified in ... exists

Value

`use_testdata()` returns `TRUE` if object was successfully saved.

`has_testdata()` returns `TRUE` if package has a '`tests/testthat/testdata`' folder.

`read_testdata()` returns a single R object

`find_testdata()` returns the normalized path to a file in a in the testdata directory

Side effects

`use_testdata()` saves an R object to a '`testdata`' dir in the current package.

See Also

`base::readRDS()`

Other infrastructure: `use_test_subdir()`, `use_tester()`

Examples

```
## Not run:
  use_testdata(letters, LETTERS)

## End(Not run)
```

---

use_tester *Use a tester function*

Description

Quickly create an R script that contains a function for running all tests in a predefined directory. This function powers the `make_tester` option of `use_test_subdir()` and you will likely not need to run it manually.

Usage

```
use_tester(
  path,
  ignore = TRUE,
  tester_path = file.path("R", "testthis-testers.R")
)
```

Arguments

`path`	`character` scalar. Name of the subdirectory of '`tests/testthat/`' for which to create a tester function.
`ignore`	`logical` scalar. Add `tester_path` to '`.Rbuildignore`'?
`tester_path`	`logical` scalar. Path to the R script file in which to store the tester functions

Value

TRUE on success (invisibly).

See Also

Other infrastructure: use_test_subdir(), use_testdata()

---

use_test_subdir *Use test subdirectories*

Description

Create a subdir in 'tests/testthat/' and optionally an R script containing a helper function to run all tests in that subdir. Useful for separating long-running tests from your unit tests, or storing tests that you do not want to run on CRAN or during R CMD Check. For running tests in 'tests/testthat/' subdirectories see test_subdir().

Usage

```
use_test_subdir(path, make_tester = TRUE, ignore_tester = TRUE)

use_integration_tests()

use_acceptance_tests()

use_manual_tests()
```

Arguments

path	character scalar. Will be processed with base::make.names() to make a syntactically valid name.
make_tester	logical or character scalar. Create an R script with a test helper function. If TRUE an R script file will be placed into the 'R/' directory of the current package, containing a function definition for running the tests in path. The file will be named 'testthis-testers.R', but you can specify your own name by passing a character scalar to make_tester(). See use_tester() for details.
ignore_tester	logical scalar'. Add 'tester' file to '.Rbuildignore'?

Value

TRUE on success (invisibly).

Test subdirectory presets

Three preset test subdirs are defined at the moment:

`test_integration()` Integration tests, also called component tests. Put tests here that test if several functions / parts of your program work together as expected. You can create the relevant subdir '`testthat/integration_tests/`' with `use_integration_tests()`.

`test_acceptance()` Acceptance tests. This is the highest levels of tests. Put tests here that verifies if your package fulfills the goals/requirements you set out to achieve with your package were met. You can create the relevant subdir '`testthat/acceptance_tests/`' with `use_acceptance_tests()`.

`test_manual()` Manual tests. Put tests here that produce output that has to be manually verified, such as: console output, pdf files, plots. It is recommended you collect the output files of such tests in '`tests/testthat/testout`'. You can create the relevant subdir with '`testthat/manual_tests/`' with `use_manual_tests()`.

You can modify the default paths for manual, acceptance and integration tests by setting the respective `options()`, but it is recommended to create your own test subdirs instead.

See Also

`test_subdir()`

Other infrastructure: `use_testdata()`, `use_tester()`

Examples

```
## Not run:
use_test_subdir("special_tests", make_tester = TRUE)

## Reload the Package manually...
## Create some tests in tests/testthat/test_special_tests/

test_special_tests()

## End(Not run)
```

Index

∗ **infrastructure**
use_test_subdir, 10
use_testdata, 8
use_tester, 9

base::make.names(), *10*
base::readRDS(), *9*

devtools::test(), *6*

find_testdata *(*use_testdata*)*, 8

get_test_coverage, 2

has_testdata *(*use_testdata*)*, 8

open_testfile, 2
options(), *3*

print.test_index *(*test_index*)*, 5

read_testdata *(*use_testdata*)*, 8

test_acceptance *(*test_subdir*)*, 6
test_all *(*test_subdir*)*, 6
test_coverage, 4
test_coverage(), *2*
test_index, 5
test_integration *(*test_subdir*)*, 6
test_manual *(*test_subdir*)*, 6
test_skeleton, 5
test_subdir, 6
test_subdir(), *10, 11*
test_this, 7
testthat::test_file(), *7*
testthat::testthat_results, *6*
testthis, 3
testthis-package *(*testthis*)*, 3

use_acceptance_tests *(*use_test_subdir*)*, 10
use_integration_tests *(*use_test_subdir*)*, 10
use_manual_tests *(*use_test_subdir*)*, 10
use_test_subdir, *9, 10*, 10
use_test_subdir(), *6, 7, 9*
use_testdata, 8, *10, 11*
use_testdata_raw *(*use_testdata*)*, 8
use_tester, *9*, 9, *11*
use_tester(), *10*
usethis::edit_r_profile(), *4*
usethis::proj_get(), *3*
usethis::proj_set(), *3*
usethis::use_covr_ignore(), *4*
usethis::use_test(), *6*