Package 'testthis'

August 11, 2018

Type Package

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

Version 1.0.4

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

Description Utility functions and 'RStudio' addins to ease the life of people using 'testthat', 'devtools' and 'usethis' in their package development workflow. Hotkeyable addins are provided for such common tasks as switching between a source file and an associated test file, or running unit tests in a single file. 'testthis' also provides utility function to manage and run tests in subdirectories of the test/testthat directory.

License MIT + file LICENSE

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

Suggests rprojroot, knitr, rmarkdown, rstudioapi, roxygen2 (>= 5.0.0)

Encoding UTF-8

RoxygenNote 6.0.1.9000

VignetteBuilder knitr

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

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

NeedsCompilation no

Author Stefan Fleck [aut, cre]

Repository CRAN

Date/Publication 2018-08-11 05:40:05 UTC

R topics documented:

`get_test_coverage` *Get Test Coverage of Package*

Description

This extracts the test coverage of the target package (usually the package you are working on). Bear in mind that testthis uses a checklist-approach for this, and depends that you either put the function name in your `test_that()` calls, or used test_this tags. If you want automatic analysis of test coverage, you must look in other packages such as `covr`.

Usage

```
get_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 testfile that contains `test_that("myfunction works", {...})`, myfunction will be marked as tested.

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 <- get_test_coverage()
as.data.frame(x)

## End(Not run)
```

`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 an R script and the associated test file.

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 `vignette("testthis")`

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...

Author(s)

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

See Also

`usethis::edit_r_profile()`

---

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

Value

A 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).

Value

`NULL` (invisibly)

test_with_skip *Execute all test_that tests in a package, except some.*

Description

This is a wrapper for `devtools::test()` that skips `test_*.R` files that contain the testthis tag `#' @skip`.

Usage

```
test_with_skip(...)
```

Arguments

`...`	additional arguments passed to `test_dir`

Details

See the devtools documentation for further info or `vignette("testthis")` for infos on testthis tags.

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)

has_testdata()

read_testdata(infile, subdir = NULL)
```

Arguments

`...`	R objects to save to the '`testdata`' dir. If empty, an empty directory is created.
`subdir`	Character scalar. Subdirectory of '`test_data`' 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.
`infile`	rds file to read (must end in .rds, otherwise .rds ending is automatically added)

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

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_testdata_raw, use_tester, use_testignore

Examples

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

## End(Not run)
```

---

use_testdata_raw *Create* testdata-raw *folder.*

Description

A folder to put scripts in that produce the files in 'testdata'

Usage

```
use_testdata_raw()
```

See Also

Other infrastructure: use_test_subdir, use_testdata, use_tester, use_testignore

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

Value

`TRUE` on success (invisibly).

See Also

Other infrastructure: `use_test_subdir`, `use_testdata_raw`, `use_testdata`, `use_testignore`

use_testignore *Tell testthis that a function does not need to be tested*

Description

Tell testthis that a function does not need to be tested (for `get_test_coverage()`)

Usage

```
use_testignore(x)
```

Arguments

`x`	a character vector of ignores (function names).

Value

TRUE (invisibly).

See Also

Other infrastructure: use_test_subdir, use_testdata_raw, use_testdata, use_tester

Examples

```
## Not run:
use_testignore("helperfunction")

## End(Not run)
```

---

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. 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_raw`, `use_testdata`, `use_tester`, `use_testignore`

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