

Package: doctest (via r-universe)

September 27, 2026

Type Package

Title Generate Tests from Examples Using 'roxygen' and 'testthat'

Version 0.4.1

Maintainer David Hugh-Jones <davidhughjones@gmail.com>

Description Creates 'testthat' tests from 'roxygen' examples using simple tags.

License MIT + file LICENSE

Encoding UTF-8

Imports cli, fs, pkgload, purrr, rlang, roxygen2, testthat

Suggests covr, knitr, lifecycle, rmarkdown

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

URL <https://hughjonesd.github.io/doctest/>

BugReports <https://github.com/hughjonesd/doctest/issues>

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libuv1-dev libxml2-dev

Repository <https://r-multiverse-staging.r-universe.dev>

Date/Publication 2026-07-29 19:16:27 UTC

RemoteUrl <https://github.com/hughjonesd/doctest>

RemoteRef 7c66bdbe6352d8339cf977f3a3d40d7c0af9a475

RemoteSha 7c66bdbe6352d8339cf977f3a3d40d7c0af9a475

Contents

doctest-package	2
doctest-tag	4
doctestExample-tag	5

dt_roclet	5
expect-tag	6
expectRaw-tag	7
omit-tag	7
snap-tag	8
test_doctests	8
testRaw-tag	9

Index	10
--------------	-----------

doctest-package	<i>Write testthat tests for your examples, using roxygen tags</i>
-----------------	---

Description

The doctest package lets you test the code in your "Examples" section in .Rd files. It uses the roxygen2 and testthat packages. For more information, see [@doctest](#) and [@expect](#).

Details

Example:

Here's some **roxygen** documentation for a function:

```
#' Fibonacci function
#'
#' @param n Integer
#' @return The nth Fibonacci number
#'
#' @doctest
#'
#' @expect type("integer")
#' fib(2)
#'
#' n <- 6
#' @expect equal(8)
#' fib(n)
#'
#' @expect warning("not numeric")
#' fib("a")
#'
#' @expect warning("NA")
#' fib(NA)
fib <- function (n) {
  if (! is.numeric(n)) warning("n is not numeric")
  ...
}
```

Instead of an @examples section, we have a @doctest section.

This will create tests like:

```
# Generated by doctest: do not edit by hand
# Please edit file in R/<text>
```

```
test_that("Doctest: fib", {
  # Created from @doctest for `fib`
  # Source file: <text>
  expect_type(fib(2), "integer")
  n <- 6
  expect_equal(fib(n), 8)
  expect_warning(fib("a"), "not numeric")
  expect_warning(fib(NA), "NA")
})
```

The .Rd file will be created as normal, with an example section like:

```
\examples{
fib(2)

n <- 6
fib(n)
fib("a")
fib(NA)
}
```

Usage:

Install doctest from **r-universe**:

```
install.packages("doctest", repos = c("https://hughjonesd.r-universe.dev",
                                         "https://cloud.r-project.org"))
```

Or from CRAN:

```
install.packages("doctest")
```

Or get the development version:

```
devtools::install("hughjonesd/doctest")
```

To use doctest in your package, alter its DESCRIPTION file to add the dt_roclet roclet and "doctest" package to roxygen:

```
Roxygen: list(roclets = c("collate", "rd", "namespace",
                          "doctest::dt_roclet"), packages = "doctest")
```

Then use roxygen2::roxygenize() or devtools::document() to build your package documentation.

Doctest is **[Experimental]**.

Author(s)

Maintainer: David Hugh-Jones <davidhughjones@gmail.com>

Authors:

- David Hugh-Jones <davidhughjones@gmail.com>

doctestExample-tag	<i>Add an example from a file</i>
--------------------	-----------------------------------

Description

`@doctestExample path/to/file.R` is a drop-in replacement for `@example path/to/file.R`. It doesn't add the contents of `file.R` to the test.

Details

If you have complex examples you may want to store them separately. Roxygen2 uses the `@example` tag for this. `@doctestExample` does the same: it adds the contents of its file to the resulting example. Suppose `man/R/example-code.R` contains the line:

```
2 + 2
```

Then the following roxygen:

```
#' @doctest
#'\n#' @expect equal(2)\n#' 1 + 1\n#' @doctestExample man/R/example-code.R
```

will generate an example like:

```
1 + 1\n2 + 2
```

At present, `@doctestExample` doesn't add any code to the tests. `@doctestExample` was added in doctest 0.3.0.

dt_roclet	<i>Create the doctest roclet</i>
-----------	----------------------------------

Description

You can use this in your package DESCRIPTION like this:

```
Roxygen: list(roclets = c("collate", "rd", "namespace", "doctest::dt_roclet"))
```

Usage

```
dt_roclet()
```

Value

The doctest roclet

Examples

```
## Not run:
roxygen2::roxygenize(roclets = "doctest::dt_roclet")

## End(Not run)
```

expect-tag	<i>Create an expectation</i>
------------	------------------------------

Description

@expect creates an expectation for your example code.

Details

Use @expect to create a [testthat](#) expectation.

```
#' @doctest
#'
#' @expect equals(4)
#' 2 + 2
#'
#' f <- function () warning("Watch out")
#' @expect warning()
#' f()
```

The next expression will be inserted as the first argument to the expect_* call.

Don't include the expect_ prefix.

If you want to include the expression in a different place or places, use a dot .:

```
@expect equals(., rev(.))
c("T", "E", "N", "E", "T")
```

The @expect tag and code must fit on a single line.

See Also

Other expectations: [expectRaw-tag](#), [snap-tag](#)

expectRaw-tag	Create an expectation as-is
---------------	-----------------------------

Description

@expectRaw creates an expectation for your example code, without adding the next expression as the subject.

Details

@expectRaw creates a [testthat](#) expectation. Unlike [@expect](#), it doesn't insert the subsequent expression as the first argument.

```
#' @doctest
#'
#' x <- 2 + 2
#' @expectRaw equals(x, 4)
#'
#' f <- function () warning("Watch out")
#' @expectRaw warning(f())
```

Don't include the expect_ prefix.

The @expectRaw tag and code must fit on a single line.

See Also

Other expectations: [expect-tag](#), [snap-tag](#)

omit-tag	Exclude example code from a test
----------	----------------------------------

Description

@omit excludes example code from a test until the next tag. Use @resume to restart including code without creating an expectation.

Details

Use @omit to avoid redundant or noisy code:

```
#' @doctest
#'
#' @expect equal(0)
#' sin(0)
#'
#' @omit
```


Arguments

path	Path to package
...	Passed to <code>testthat::test_local()</code> .

Value

The result of `testthat::test_local()`.

Examples

```
## Not run:
  test_doctests()

## End(Not run)
```

testRaw-tag	<i>Add a line of code to the test</i>
-------------	---------------------------------------

Description

`@testRaw` adds an arbitrary line of code to your test, without including it in the .Rd example.

Details

`@testRaw` adds an arbitrary line of code to your test:

```
#' @doctest
#' @testRaw skip_on_cran("Takes too long")
#' @expect equal(6765)
#' fibonacci(20)
```

Unless your doctest has at least one `@expect` or `@expectRaw` tag, no test will be created. So use those tags, not `@testRaw`, to add expectations.

Remember that the main purpose of examples is to document your package for your users. If your code is getting too different from your example, consider splitting it off into a proper test file. To do this, rename the file in `tests/testthat`, and deleting the Generated by doctest comment.

Index

* expectations

- expect-tag, [6](#)
- expectRaw-tag, [7](#)
- snap-tag, [8](#)

- doctest (doctest-package), [2](#)
- doctest-package, [2](#)
- doctest-tag, [4](#)
- doctestExample-tag, [5](#)
- dt_roclet, [5](#)

- expect-tag, [6](#)
- expectRaw-tag, [7](#)

- omit-tag, [7](#)

- resume-tag (omit-tag), [7](#)

- snap-tag, [8](#)

- test_doctests, [8](#)
- testRaw-tag, [9](#)
- testthat, [4](#), [6](#), [7](#)
- testthat::test_local(), [8](#), [9](#)