

Package ‘doctest’

January 11, 2024

Type Package

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

Version 0.3.0

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

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

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

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

Suggests covr, knitr, lifecycle, rmarkdown

Config/testthat/edition 3

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

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

VignetteBuilder knitr

NeedsCompilation no

Author David Hugh-Jones [aut, cre]

Repository CRAN

Date/Publication 2024-01-11 22:10:02 UTC

R topics documented:

doctest-package	2
doctest-tag	4
doctestExample-tag	5
dt_roclet	5
expect-tag	6
expectRaw-tag	7
omit-tag	7
snap-tag	8
testRaw-tag	8
test_doctests	9

doctest-package *Write testthat tests for your examples, using roxygen tags*

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>
```

```

# Source line: 7
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>

See Also

Useful links:

- <https://hughjonesd.github.io/doctest/>
- Report bugs at <https://github.com/hughjonesd/doctest/issues>

doctest-tag

Start a doctest

Description

`@doctest` starts a doctest: a code example that also contains one or more [testthat](#) expectations.

Details

Use `@doctest` where you would usually use `@examples`. Then add [@expect](#) and [@expectRaw](#) tags beneath it to create expectations.

By default, a test labelled "Example: <object name>" is created. You can put a different label after `@doctest`:

```
#' @doctest Positive numbers
#'
#' x <- 1
#' @expect equal(x)
#' abs(x)
#'
#' @doctest Negative numbers
#' x <- -1
#' @expect equal(-x)
#' abs(x)
```

You can have more than one `@doctest` tag in a roxygen block. Each doctest will create a new test, but they will all be merged into a single Rd example. Each doctest must contain an independent unit of code. For example, this won't work:

```
#' @doctest Test x
#' @expect equal(2)
#' x <- 1 + 1
#'
#' @doctest Keep testing x
#' @expect equal(4)
#' x^2
#' # Test will error, because `x` has not been defined here
```

A test will only be written if the `@doctest` section has at least one [@expect](#) or [@expectRaw](#) in it. This lets you change `@examples` to `@doctest` in your code, without generating unexpected tests.

doctestExample-tag *Add an example from a file*

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

will generate an example like:

```
1 + 1
2 + 2
```

At present, `@doctestExample` doesn't add any code to the tests. `@doctestExample` was added after doctest 0.2.0.

dt_roclet *Create the doctest roclet*

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

Create an expectation

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	<i>Create an expectation as-is</i>
---------------	------------------------------------

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	<i>Exclude example code from a test</i>
----------	---

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
```

```
#' curve(sin(x), 0, 2 * pi)
#
#' @expect equal(1)
#' cos(0)
```

@omit is separate from \donttest and \dontrun tags in Rd files. This allows you to test code that would cause an error if run by R CMD CHECK. If you also want R CMD CHECK to skip your code, you should use \donttest{} separately (see [writing R extensions](#)).

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. You can do this by renaming it and deleting the Generated by doctest comment.

snap-tag

Create a snapshot test

Description

@snap creates a **snapshot test** for your example. It is shorthand for @expect snapshot().

Details

Often, examples show complex output to the user. If this output changes, you want to check that it still "looks right". Snapshot tests help by failing when the output changes, and allowing you to review and approve the new output.

```
#' @doctest
#
#' @snap
#' summary(lm(Petal.Width ~ Species, data = iris))
```

See Also

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

testRaw-tag

Add a line of code to the test

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.

test_doctests	<i>Test doctests in a package</i>
---------------	-----------------------------------

Description

This is a utility function to run doctests in a local source package. It calls `testthat::test_local()`.

Usage

```
test_doctests(path = ".", ...)
```

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)
```

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