

# Package ‘exampletestr’

April 23, 2018

**Type** Package

**Title** Help for Writing Unit Tests Based on Function Examples

**Version** 1.3.0

**Maintainer** Rory Nolan <rorynolan@gmail.com>

**Description** Take the examples written in your documentation of functions and use them to create shells (skeletons which must be manually completed by the user) of test files to be tested with the 'testthat' package.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** filesstrings (>= 2.2.0), magrittr, stringr, devtools, styler, purrr, roxygen2, checkmate, readr, utils, ore, rprojroot

**RoxygenNote** 6.0.1

**URL** <https://www.github.com/rorynolan/exampletestr>

**BugReports** <https://www.github.com/rorynolan/exampletestr/issues>

**Suggests** testthat, covr, knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Rory Nolan [aut, cre],  
Sergi Padilla-Parra [ths]

**Repository** CRAN

**Date/Publication** 2018-04-23 10:55:37 UTC

## R topics documented:

construct_expect_equal . . . . .	2
extract_examples . . . . .	2
extract_examples_rd . . . . .	3
extract_expressions . . . . .	4

make_test_shell . . . . .	5
test-shells . . . . .	6
text_parse_error . . . . .	7

<b>Index</b>	<b>8</b>
--------------	----------

---

construct\_expect\_equal  
*Construct an expect\_equal expression*

---

### Description

Construct an expect\_equal expression from a character vector containing an expression to be evaluated.

### Usage

```
construct_expect_equal(text_expr)
```

### Arguments

text\_expr      A character vector of lines that, when executed produce a single output.

### Value

A character vector. The lines of text containing the expect\_equal code corresponding to the input, which will help to write the test file based on documentation examples. Remember that this is something that you're intended to fill the gaps in later.

### Examples

```
text_expr <- c("sum(1, ", "2)")
cat(paste(text_expr, collapse = "\n"))
construct_expect_equal(text_expr)
cat(paste(construct_expect_equal(text_expr), collapse = "\n"))
```

---

extract\_examples      *Extract examples lines from the functions in a .R file of a package.*

---

### Description

In each .R file in the R/ folder of a package project, for the functions defined therein, there can corresponding examples in the .Rd files of the man/ folder. This function extracts those examples into a list of character vectors, one list element for each documented function.

### Usage

```
extract_examples(r_file_name, pkg_dir = ".", document = TRUE)
```

**Arguments**

<code>r_file_name</code>	The name of the <code>.R</code> file within <code>R/</code> . There's no need to specify the file path (as <code>R/x.R</code> , but you can do this if you want), you can just use <code>x.R</code> for whichever file <code>x</code> it is. You can also omit the <code>.R</code> for convenience, however using the wrong case (e.g. <code>.r</code> ) will produce an error.
<code>pkg_dir</code>	The directory of the R project for this package (defaults to current directory). This is the parent directory of <code>R/</code> and <code>man/</code> . In reality, this specification is somewhat lenient and even if you are in any sub-directory of the root, it will work (e.g. it will work if you are in <code>R/</code> or <code>man/</code> ). Beware that this behaviour can cause a problem if you have an R package inside an R package (but really, you have yourself to blame if that's the case).
<code>document</code>	Run <code>devtools::document()</code> to update package documentation before starting?

**Details**

Anything found within a `\dontrun{...}` block is ignored.

**Value**

A list of character vectors.

**Examples**

```
devtools::create("tempkg")
setwd("tempkg")
file.copy(system.file("extdata", "detect.R", package = "exampletestr"), "R")
devtools::document()
exampletestr::extract_examples("detect")
setwd("../")
filesstrings::dir.remove("tempkg")
```

---

`extract_examples_rd`     *Extract examples from a .Rd file as a character vector.*

---

**Description**

This is a convenient wrapper to `tools::Rd2ex` which actually returns a character vector of the examples in the `.Rd` file.

**Usage**

```
extract_examples_rd(rd_file_path)
```

**Arguments**

`rd_file_path`     The path to the `.Rd` file.

**Value**

A character vector.

**Examples**

```
this_function_rd <- system.file("extdata", "str_detect.Rd",  
                                package = "exampletestr")  
extract_examples_rd(this_function_rd)
```

---

extract\_expressions    *Text expression groups.*

---

**Description**

Given a character vector of R expressions, break the vector up into groups of lines, where each group of lines is a valid R expression.

**Usage**

```
extract_expressions(text_expr)
```

**Arguments**

text\_expr    A character vector.

**Value**

A list of character vectors, each of which can be evaluated as a valid R expression.

**Examples**

```
text_expr <- c("a <- 1",  
              "fx <- function(x) {",  
              "  x + 1",  
              "} # this comment will disappear")  
extract_expressions(text_expr)
```

---

make_test_shell	<i>Make the shell of a test_that test.</i>
-----------------	--

---

## Description

Given a character vector of the examples from a function, create the shell of a `testthat::test_that()` code block (to be filled in by the user) based upon those examples.

## Usage

```
make_test_shell(example_block, desc = "", e_e = TRUE)
```

## Arguments

<code>example_block</code>	A character vector of the lines in the examples of a function's documentation.
<code>desc</code>	To be the <code>desc</code> argument of the <code>testthat::test_that()</code> call.
<code>e_e</code>	Set this to <code>FALSE</code> to prevent anything from being put in the shell of an <code>expect_equal()</code> statement.

## Details

Assignment lines (lines with `<-`, or even an `=` assignment (naughty, I know)) and lines starting with `print()`, `stop()`, `warning()`, `setwd()`, `plot()`, `ggplot()`, `set.seed` or `library()` are left alone, others are put in the shell of an `expect_equal()` statement. To prevent anything from being put in the shell of an `expect_equal()` statement, set `e_e = FALSE`. Anything found within a `\dontrun{...}` block is ignored.

## Value

A character vector giving the shell of a `test_that` function call testing all of the calls in the example block.

## Examples

```
devtools::create("tempkg")
setwd("tempkg")
file.copy(system.file("extdata", "detect.R", package = "exampletestr"), "R")
devtools::document()
make_test_shell(extract_examples("detect")[[1]])
make_test_shell(extract_examples("detect")[[1]],
                desc = "xyz", e_e = FALSE)
setwd("../")
filesstrings::dir.remove("tempkg")
```

test-shells

*Create test shells.***Description**

- For a given function `fun()` in a package, `make_test_shell_fun()` checks if there are examples for that function detailed in the `man/` directory (in a `.Rd` file) and if so creates a shell (skeleton) of a `testthat::test_that()` test based on those examples via `make_test_shell()`. The created shell is then written to a corresponding file `test-fun-examples.R` in `tests/testthat`.
- For a given file `x.R` in the `R/` directory of a package, for each function defined in that `.R` file, `make_tests_shells_file()` checks if there are examples for that function detailed in the `man/` directory (in a `.Rd` file) and if so creates a shell (skeleton) of a `testthat::test_that()` test based on those examples via `make_test_shell()`. The created shells are then written to a corresponding file `test-x-examples.R` in `tests/testthat`.
- `make_test_shells_pkg()` runs `make_test_shells_file()` on every `.R` file in the `R/` directory of a package.

**Usage**

```
make_test_shell_fun(fun, pkg_dir = ".", overwrite = FALSE, e_e = TRUE,
  open = TRUE, document = TRUE)
```

```
make_tests_shells_file(r_file_name, pkg_dir = ".", overwrite = FALSE,
  e_e = TRUE, open = TRUE, document = TRUE)
```

```
make_tests_shells_pkg(pkg_dir = ".", overwrite = FALSE, e_e = TRUE,
  open = FALSE, document = TRUE)
```

**Arguments**

<code>fun</code>	The name of the function to make a test shell for.
<code>pkg_dir</code>	The directory of the R project for this package (defaults to current directory). Note that this is the parent directory of <code>R/</code> .
<code>overwrite</code>	Overwrite if the test file you're trying to create already exists?
<code>e_e</code>	Set this to <code>FALSE</code> to prevent anything from being put in the shell of an <code>expect_equal()</code> statement.
<code>open</code>	Open the created test file in your editor after it is created?
<code>document</code>	Run <code>devtools::document()</code> to update package documentation before starting?
<code>r_file_name</code>	The name of the <code>.R</code> file within <code>R/</code> . There's no need to specify the file path (as <code>R/x.R</code> , but you can do this if you want), you can just use <code>x.R</code> for whichever file <code>x</code> it is. You can also omit the <code>.R</code> for convenience, however using the wrong case (e.g. <code>.r</code> when the file actually has the extension <code>.R</code> ) will produce an error.

**Value**

The shell of the test file is written into tests/testthat. It has the same name as the .R file it was created from except it has "test\_" tacked onto the front.

**Examples**

```
devtools::create("tempkg")
setwd("tempkg")
file.copy(system.file("extdata", "detect.R", package = "exampletestr"), "R")
make_test_shell_fun("str_detect()", document = TRUE, open = FALSE)
make_tests_shells_file("detect", document = FALSE, open = FALSE)
make_tests_shells_pkg(overwrite = TRUE, document = FALSE)
setwd("../")
filesstrings::dir.remove("tempkg")
```

---

text_parse_error	<i>Does parsing of text give an error?</i>
------------------	--

---

**Description**

Can a character vector (where each line is treated as a line of R code) be parsed as an R expression (or several R expressions) without giving an error?

**Usage**

```
text_parse_error(text_expr)
```

**Arguments**

text\_expr      The expression to be evaluated, as a character vector.

**Value**

TRUE if the code gives an error and FALSE otherwise.

**Examples**

```
text_parse_error("a <- 1")
text_parse_error("a <- ")
```

# Index

`construct_expect_equal`, 2

`devtools::document()`, 3, 6

`extract_examples`, 2

`extract_examples_rd`, 3

`extract_expressions`, 4

`make_test_shell`, 5

`make_test_shell()`, 6

`make_test_shell_fun (test-shells)`, 6

`make_test_shell_fun()`, 6

`make_test_shells_file()`, 6

`make_test_shells_pkg()`, 6

`make_tests_shells_file (test-shells)`, 6

`make_tests_shells_file()`, 6

`make_tests_shells_pkg (test-shells)`, 6

`test-shells`, 6

`testthat::test_that()`, 5, 6

`text_parse_error`, 7

`tools::Rd2ex`, 3