Package ‘patrick’

August 13, 2018

Title Parameterized Unit Testing

Author Michael Quinn <`msquinn@google.com`>

Maintainer Michael Quinn <`msquinn@google.com`>

Description This is an extension of the 'testthat' package that lets you add parameters to your unit tests. Parameterized unit tests are often easier to read and more reliable, since they follow the DNRY (do not repeat yourself) rule.

Version 0.0.1

Depends R (>= 3.1)

Imports dplyr, purrr, rlang, testthat, tibble

License Apache License 2.0

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-08-13 14:10:03 UTC

R topics documented:

patrick 1
with_parameters_test_that 2

Index **4**

`patrick` *Parameterized Unit Testing*

Description

`patrick` (parameterized testing in R is kind of cool!) is a `testthat` extension that lets you create reusable blocks of a test codes. Parameterized tests are often easier to read and more reliable, since they follow the DNRY (do not repeat yourself) rule.

Details

This package is inspired by parameterized testing packages in other languages, notably the parameterized library in Python.

`with_parameters_test_that`

Execute a test with parameters.

Description

This function is an extension of `testthat::test_that()` that lets you pass a series of testing parameters. These values are substituted into your regular testing code block, making it reusable and reducing duplication.

Usage

```
with_parameters_test_that(desc_stub, code, .cases = NULL, ...)

cases(...)
```

Arguments

`desc_stub`	A string scalar. Used in creating the names of the parameterized tests.
`code`	Test code containing expectations.
`.cases`	A data frame where each row contains test parameters.
`...`	Named arguments of test parameters.

Details

You have a couple of options for passing parameters to you test. You can use named vectors/lists. The function will assert that you have correct lengths before proceeding to test execution. Alternatively you can used a `data.frame` or list in combination with the splice unquote operator `!!!`. Last, you can use the constructor `cases()`, which is similar to building a `data.frame` rowwise. If you manually build the data frame, pass it in the `.cases` argument.

One parameter is noteworthy. If the user passes a character vector as `test_name`, each instance is combined with `desc_stub` to create the completed test name. Similarly, the named argument from `cases()` is combined with `desc_stub` to create the parameterized test names.

Examples

```
with_parameters_test_that("trigonometric functions match identities", {
    testthat::expect_equal(expr, numeric_value)
  },
  expr = c(sin(pi / 4), cos(pi / 4), tan(pi / 4)),
  numeric_value = c(1 / sqrt(2), 1 / sqrt(2), 1)
)
```

```
# Run the same test with the cases() constructor
with_parameters_test_that("trigonometric functions match identities", {
    testthat::expect_equal(expr, numeric_value)
  },
  cases(
    sin = list(expr = sin(pi / 4), numeric_value = 1 / sqrt(2)),
    cos = list(expr = cos(pi / 4), numeric_value = 1 / sqrt(2)),
    tan = list(expr = tan(pi / 4), numeric_value = 1)
  )
)
```

Index

cases *(*with_parameters_test_that*)*, 2

patrick, 1
patrick-package *(*patrick*)*, 1

testthat::test_that(), *2*

with_parameters_test_that, 2