What are unit tests and why write them?

DEVELOPING R PACKAGES

Aimée Gott

Education Practice Lead, Mango Solutions

Why write unit tests?

A function that works correctly now may not behave as expected in the future if:

- Supporting or connected code could be added or modified
- A later version of R and/or later versions of packages are used
- The code is run on new data
- The code is run on a different operating system

Setting up the test structure

- Call `use_testthat` to set up the test framework
- This creates a `test` directory in the package root directory
- Within the `test` directory, there is a script `testthat.R` which contains code to run the tests
- Within the `test` directory is a directory `testthat` where you save all of your test scripts

Writing an individual test

Some of the most common expect statements:

- `expect_identical` - Checks for exact equality
- `expect_equal` - Checks for equality with numerical tolerance
- `expect_equivalent` - More relaxed version of equals
- `expect_error` - Checks that an expression throws an error
- `expect_warning` - Checks that an expression gives a warning
- `expect_output` - Checks that output matches a specified value

expect_identical

- Strictest numerical comparison
- Compares **values**, **attributes**, and **type**

Passes

```
library(testthat)

my_vector <- c("First" = 1, "Second" = 2)

expect_identical(my_vector, c("First" = 1, "Second" = 2))
```

expect_identical

Fails

```
expect_identical(myvector, c(1, 2))
```

```
Error: `vec1` not identical to c(1, 2).
names for target but not for current
```

expect_equal

- Compares **values** and **attributes**
- Doesn't compare **type**

Passes

```
expect_equal(my_vector, c("First" = 1L, "Second" = 2L))
```

- Can set `tolerance` parameter to allow for small differences
- Only differences larger than the `tolerance` value will cause the test to fail

Fails

```
expect_equal(my_vector, c(First = 1.1, Second = 2.1))
```

```
Error: `my_vector` not equal to c(First = 1.1, Second = 2.1).
2/2 mismatches (average diff: 0.1)
[1] 1 - 1.1 == -0.1
[2] 2 - 2.1 == -0.1
```

Passes

```
expect_equal(my_vector, c(First = 1.1, Second = 2.1), tolerance = 0.1)
```

expect_equivalent

- Least strict numerical comparison
- Compares **values** only
- Doesn't compare **attributes** or **type**

Passes

```
expect_equivalent(my_vector, c(1, 2))
```

Let's practice!

DEVELOPING R PACKAGES

Testing errors and warnings

DEVELOPING R PACKAGES

Nic Crane

Data Science Consultant, Mango Solutions

Testing errors and warnings

Warning

```
sqrt(-1)
```

```
NaN
Warning message:
In sqrt(-1) : NaNs produced
```

Error

```
sqrt("foo")
```

```
Error in sqrt("foo") : non-numeric argument to mathematical function
```

Testing warnings

Passes

```
expect_warning(sqrt(-1))
```

Testing errors

Passes

```
expect_error(sqrt("foo"))
```

Fails

```
expect_error(sqrt(-1))
```

```
Error: sqrt(-1) did not throw an error.
In addition: Warning message:
In sqrt(-1) : NaNs produced
```

Testing specific warning and error messages

Passes

```
expect_error(sqrt("foo"), "non-numeric argument to mathematical function")
```

Fails

```
expect_error(sqrt("foo"), "NaNs produced")
```

```
Error: error$message does not match "NaNs produced".
Actual value: "non-numeric argument to mathematical function"
```

Let's practice!

DEVELOPING R PACKAGES

Testing specific output and non-exported functions

DEVELOPING R PACKAGES

Aimée Gott

Education Practice Lead, Mango Solutions

Testing specific output

```
str(airquality)
```

```
'data.frame':    153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

Testing for expected output

Passes

```
expect_output(str(airquality), "41 36 12 18 NA 28 23 19 8 NA")
```

Fails

```
expect_output(str(airquality), "air")
```

```
Error: str(airquality) does not match "air".
Actual value: "'data.frame':\t153 obs. of  6 variables:\n $ Ozone   : int
41 36 12 18 NA 28 23 19 8 NA ...\n $ Solar.R: int  190 118 149 313 NA NA
299 99 19 194 ...\n $ Wind    : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8
20.1 8.6 ...\n $ Temp    : int  67 72 74 62 56 66 65 59 61 69 ...\n $
Month   : int  5 5 5 5 5 5 5 5 5 5 ...\n $ Day     : int  1 2 3 4 5 6 7 8 9
10 ..."
```

Testing for expected output from a file

First run - create file

```
expect_output_file(str(airquality), "airq.txt", update = TRUE)
```

```
Error: str(airquality) not equal to safe_read_lines("airq.txt").
Lengths differ: 7 vs 0
In addition: Warning messages:
1: In file(con, "r") :
  cannot open file 'airq.txt': No such file or directory
2: In value[[3L]](cond) : cannot open the connection
```

Subsequent runs - comparing to file

```
expect_output_file(str(airquality), "airq.txt")
```

Testing exported functions example

```
expect_equivalent(na_counter(airquality),
                  c(37, 7, 0, 0, 0, 0))
```

Testing non-exported functions

Fails

```
expect_equal(sum_na(airquality$Ozone), 37)
```

```
Error in compare(object, expected, ...) :
    could not find function "sum_na"
```

Passes

```
expect_equal(simutils:::sum_na(airquality$Ozone), 37)
```

Let's practice!

DEVELOPING R PACKAGES

Grouping tests and execution output

DEVELOPING R PACKAGES

Nic Crane

Data Science Consultant, Mango Solutions

Context and test_that

```
test_that("na_counter correctly counts NA values", {

  test_matrix = matrix(c(NA, 1, 4, NA, 5, 6), nrow = 2)

  air_expected = c(Ozone = 37, Solar.R = 7, Wind = 0,
                   Temp = 0, Month = 0, Day = 0)

  mat_expected = c(V1 = 1, V2 = 1, V3 = 0)

  expect_equal(na_counter(airquality), air_expected)

  expect_equal(na_counter(test_matrix), mat_expected)

})
```

Context

```
context("na_counter checks")

test_that("na_counter correctly counts NA values", {

  test_matrix = matrix(c(NA, 1, 4, NA, 5, 6), nrow = 2)

  air_expected = c(Ozone = 37, Solar.R = 7, Wind = 0,
                   Temp = 0, Month = 0, Day = 0)

  mat_expected = c(V1 = 1, V2 = 1, V3 = 0)

  expect_equal(na_counter(airquality), air_expected)
  expect_equal(na_counter(test_matrix), mat_expected)

})

test_that("na_counter returns error if data is wrong object type", {

  expect_error(na_counter(c(1, 2, 3, NA)))

})
```

Executing unit tests

```
test("simutils")
```

```
Loading simutils
Loading required package: testthat
Testing simutils
na_counter checks: ...
sample_from_data tests: ...

DONE ================================================
```

Testing during the check process

```
* checking tests ...
  Running 'testthat.R'
Warning message:
running command '"C:/PROGRA~1/R/R-34~1.2/bin/x64/R" CMD BATCH --vanilla
"testthat.R" "testthat.Rout"' had status 1
 ERROR
Running the tests in 'tests/testthat.R' failed.
Last 13 lines of output:
  > library(simutils)
  > test_check("simutils")
  1. Failure: sample_from_data returns correct output
  (@test-sample_from_data.R#11)
  df$Ozone not equal to c(22, 47, 45, 80, 7, 21, 23, 23, 16, 44).
  1/10 mismatches
  [10] 45 - 44 == 1

  testthat results ========================================================
  OK: 4 SKIPPED: 0 FAILED: 1
```

Understanding a failing test

Repeat until all test pass:

- Identify the cause
- Determine whether it's the test or the function that needs updating
- Fix your code!
- Run tests again

Let's practice!

DEVELOPING R PACKAGES

Wrap-up

DEVELOPING R PACKAGES

Aimée Gott and Nic Crane

Data Science Consultants, Mango Solutions

Chapter 1

- Structure of an R package
- DESCRIPTION file
- NAMESPACE file

Chapter 2

- Documenting your package
- Creating Roxygen headers
- Exported and non-exported functions
- Other documentation

Chapter 3

- Why checks are important
- Package dependencies
- Building packages with continuous integration

Chapter 4

- Unit tests
- The test structure
- Testing for numerical similarity
- Testing for error and warning messages
- Testing for specific output
- Testing non-exported functions
- Running tests

Congratulations!

DEVELOPING R PACKAGES