

Introduction to testing R code

[https://stirlingcodingclub.github.io/code_testing/
testing_notes.html](https://stirlingcodingclub.github.io/code_testing/testing_notes.html)

Brad Duthie

1 May 2019

Introduction: Why bother testing your code?

Good question. I need to flesh this out a bit.

Testing all of your code at once is satisfying

```
==> devtools::test()

Loading GMSE
Testing GMSE
✓ | OK F W S | Context
✓ | 65      | gmse_apply tests [2.6 s]
✓ | 2       | Agent initialisation
✓ | 2       | Cost array initialisation
| 0       | Main gmse function[1] "Initialising simulations ... "
| 1       | Main gmse function[1] "Initialising simulations ... "
| 2       | Main gmse function[1] "Initialising simulations ... "
| 3       | Main gmse function[1] "Initialising simulations ... "
| 4       | Main gmse function[1] "Initialising simulations ... "
| 6       | Main gmse function[1] "Initialising simulations ... "
| 7       | Main gmse function[1] "Initialising simulations ... "
| 8       | Main gmse function[1] "Initialising simulations ... "
| 9       | Main gmse function[1] "Initialising simulations ... "
✓ | 11      | Main gmse function [4.5 s]
✓ | 2       | Interaction array and table initialisation
✓ | 4       | Landscape initialisation
✓ | 7       | Resource initialisation
✓ | 4       | Action array initialisation
✓ | 11      | Manager model
✓ | 8       | Observation model
✓ | 6       | Resource model
| 0       | Summary functions[1] "Initialising simulations ... "
| 1       | Summary functions[1] "Initialising simulations ... "
| 2       | Summary functions[1] "Initialising simulations ... "
| 3       | Summary functions[1] "Initialising simulations ... "
| 4       | Summary functions[1] "Initialising simulations ... "
| 5       | Summary functions[1] "Initialising simulations ... "
✓ | 6       | Summary functions [3.0 s]
✓ | 11      | User model
✓ | 4       | Action and cost layer initialisation

== Results ==
Duration: 10.3 s

OK:      143
Failed:  0
Warnings: 0
Skipped: 0
```

Getting started: install the testthat package

Can install `testthat` from CRAN.

```
install.packages("testthat")
```

Or install from GitHub with the `devtools` R package.

```
devtools::install_github("r-lib/testthat");
```

Load `testthat` into Rstudio just like any other R package.

```
library(testthat);
```

Two simple functions to be tested

Consider one R script (file with .R extension) with functions.

Function 1: converts a temperature from Fahrenheit to Celsius.

```
F_to_C <- function(F_temp){  
  C_temp <- (F_temp - 32) * 5/9;  
  return(C_temp);  
}
```

Two simple functions to be tested

Consider one R script (file with .R extension) with functions.

Function 1: converts a temperature from Fahrenheit to Celsius.

```
F_to_C <- function(F_temp){  
  C_temp <- (F_temp - 32) * 5/9;  
  return(C_temp);  
}
```

Function 2: converts from Celsius to Fahrenheit.

```
C_to_F <- function(C_temp){  
  F_temp <- (C_temp * 9/5) + 32;  
  return(F_temp);  
}
```

Two simple functions to be tested

Consider one R script (file with .R extension) with functions.

Function 1: converts a temperature from Fahrenheit to Celsius.

```
F_to_C(50)
```

```
## [1] 10
```

Two simple functions to be tested

Consider one R script (file with .R extension) with functions.

Function 1: converts a temperature from Fahrenheit to Celsius.

```
F_to_C(50)
```

```
## [1] 10
```

Function 2: converts from Celsius to Fahrenheit.

```
C_to_F(10)
```

```
## [1] 50
```


How the test_that function works

Example of a testthat R script, < test-temp_conversion.R >

```
library(testthat);
context("Temperature function testing");
source("temp_conversion.R"); # Functions to test

test_that("Fahrenheit to Celsius", {

  temp_C <- F_to_C(50);

  # Test that the result is numeric
  expect_that( is.numeric(temp_C), equals(TRUE) );

  # Test that the result is the correct value
  expect_that( temp_C, equals(10) );
})
```

How to run many tests quickly

Check Rstudio is set to the same directory as the test R script(s)

```
test_dir("."); # Runs all of your tests
```

The above will run all files with the prefix test- and extension .R.

How to run many tests quickly

Check Rstudio is set to the same directory as the test R script(s)

```
test_dir("."); # Runs all of your tests
```

The above will run all files with the prefix test- and extension .R.

Test output looks like this:

```
✓ | OK F W S | Context
✗ | 4 1 | Temperature function testing | 4 | Temperature function testing

test-temp_conversion.R:30: failure: Fahrenheit to Celsius wrong
`x` not equal to `expected`.
1/1 mismatches
[1] 10 - 2 == 8

== Results ==
OK: 4
Failed: 1
Warnings: 0
Skipped: 0
```

Testing as part of your coding work flow

Hadley Wickham also makes several key points about testing

- ▶ Helps spot bugs in the code earlier and see what needs fixing
- ▶ Multiple unit tests encourages smaller, manageable functions
- ▶ Writing failing tests can be useful when fixing bugs

Testing as part of your coding work flow

Hadley Wickham also **makes several key points about testing**

- ▶ Helps spot bugs in the code earlier and see what needs fixing
- ▶ Multiple unit tests encourages smaller, manageable functions
- ▶ Writing failing tests can be useful when fixing bugs

Additional resources

- ▶ [Example of unit testing R code with testthat](#) (John D. Cook)
- ▶ [Testing R packages](#) (Hadley Wickam)