

Unit testing in R using testthat library Exercises



testthat is a testing framework developed by Hadley Wickham, which makes unit testing easy for developers.

Test scripts developed can be re-run after debugging or making changes to the functions without the hassle of developing the code for testing again.

testthat has a hierarchical structure made up of expectations, tests and contexts.

Visit this [link](#) to know more.

You should be familiar with creation of functions in R to know how this testing framework works.

Answers to the exercises are available [here](#).

Exercise 1

Install and load the package **testthat** using the appropriate function.

Exercise 2

`expect_that()` is the function that makes the binary assertion of whether or not the value is as expected.

`expect_that(x,equals(y))` reads as “it is expected that ‘a’ will be equal to ‘b’”.

Use this function to see if 5×2 equals 10



Learn more about Hadley Wickhams usefull packages in the online course [R Data Pre-Processing & Data Management – Shape your Data!](#). In this course you will learn how to work with:

- tidyrr, cleaning your data
- dplyr, shape your data
- And much more

Exercise 3

The function `equals()` checks for equality with a numerical tolerance. Let's see what that tolerance level is

Use appropriate function to see if $5*2$ equals $10 + (1e-7)$.

Does the test fail?

If no, change the value to $1e-6$ and see what happens.

Exercise 4

To exactly match the values `is_identical_to()` can be used instead of `equals()`

Using the appropriate function, check if $2*2$ is identical to $4 + (1e-8)$

Please check the documentation of this package to learn more about the available functions.

Exercise 5

Let us create a function `m()` to multiply two numbers (two arguments) and check if it throws an error with character input arguments.

Check if `m("2","3")` throws an error "non-numeric argument to binary operator"

Exercise 6

Now that we know how to check for expectations, let us create

tests.

Test is a collection of expectations, where these expectations test a single item of the functionality of a process.

`test_that()` is the function that encapsulates the description and the code to test each expectation.

The first argument is the description and the second argument is a collection of expectations.

Create a test for function 'm' with description "Testing multiplication function" and add a few scenarios to it.

1. Check if `m(2,3)` equals 6
2. Check if `m(2,c(2,3))` equals `c(4,6)`
3. Check if `m(2,"3")` throws an error "non-numeric argument to binary operator"

Exercise 7

The User can write his own expectation using the `expect()` function. This expectation should compare the input value and the expectation and report the result.

The syntax to write one is as below.

```
custom_expectation <- function() {function(x)
{expectation(condition, "Failure message")}}
```

Now, write an expectation `is_greater_10()` to check if a number is greater than 10

Exercise 8

Use the expectation defined above to check if 9 is greater than 10.

Exercise 9

tests can be put together in a file and run at once. Write tests of your choice and save them in a file.

Use the function `test_file()` to run all the tests in the file.

Exercise 10

Test files in a directory can be run at once using the function `test_dir()`.

Create multiple test files and save them in a directory. Run all the tests at once using the function.