EXPERIMENT Report

# EXPERIMENT

|  |  |  |
| --- | --- | --- |
| Report Date | Project Name | Prepared By |
| September 20, 2019 | Discrete Biostatistics | Justin Clifton |

# QUESTION: What do we want to figure out?

How unit testing works in R.

# PROCESS: WHAT WILL WE DO TO ANSWER THE QUESTION?

I will attempt to write tests for the censor function using the testthat package.

# RESULTS: WHAT HAPPENED IN THE PROCESS?

The first thing you need to do is create a test directory. To do this, create a file that starts with the word test. For example, my test file is named ‘tests’. Inside of this file, will be your testing scripts. In this script, you need to write source(‘Directory of file you want to test’). Next, you have to use the test\_that() function. Inside of this function you write the specific test that you want. For example:

source('C:/Users/Justin/Desktop/Work/ArmitageDoll.R')

test\_that("Addition is commutative",{

expect\_equal(3+7,7+3)

})

Here we are testing that addition has the commutative property. So we have made an assertion using the expect\_equal() function. This function indicates that we expect 3+7 and 7+3 to be equal. Now, in order to run these tests you now have two options. You could use the test\_file() function to test a specific file, or you could use test\_dir() to run all the test files contained in a directory. Inside of these functions, you want to pass in the pathway to your test file. **IMPORTANT NOTICE:** If you find yourself having a problem using these functions, try using single quotes instead of double quotes. You could be having a problem related to “fancy quotes”.

Here’s an example of using these functions:

test\_file("C:/Users/Justin/Documents/R Projects/tests/test\_Armitage.R")

test\_dir('C:/Users/Justin/Documents/R Projects/tests')

Next, I began running tests. The first test that I decided to start with was to see if censor was producing the correct values. I created a dataset of numbers 1-10. The numbers I used were 1, 1.5 , 1.6 , 6 , 6 , 7.9 , 8.3 ,

8.4, 8.8 ,8.5 ,9 ,10. Next I calculated the midpoints and the counts that I expected the function to return. These were

midpoints = c(0.5,1.5,5.5,7.5,8.5,9.5)

counts = c(1,2,2,3,3,1)

Next I called the function to create the df object and I used expect\_equal() to test if my function produced the results I expected. Testthat produced a message indicating that all tests were passed. Good result! Next I attempted to test if censor was right inclusive. To do this, I created decimal values in my simulated dataset.

data = c(1, 1.5 , 1.6 , 6 , 6 , 7.9 , 8.3 ,

8.4, 8.8 ,8.5 ,9 ,10)

These were the values I expected, and that my function would be tested against.

midpoints = c(0.5,1.5,5.5,7.5,8.5,9.5)

counts = c(1,2,2,1,5,1)

This test passed, indicating that our censor function is right inclusive.

Next, I wrote a test to test whether datapoints greater than 80 were being excluded from our data frame. First I generated a dataset using the seq() function.

data = seq(0.1, 100, 0.1)

After that, I calculated the expected return that will be used as our test.

midpoints = seq(5,75,10)

counts = c(100,100,100,100,100,100,100,100)

expect\_equal() function used to test for equality.

expect\_equal(midpoints,censor\_df$mids)

expect\_equal(counts,censor\_df$counts)

The test was passed, indicating that datapoints greater than 80 are being filtered.

The next test we did was to see what would happen if all datapoints fall into the same bin.

data <- seq(10.01,20,0.01)

Predicted mid points and counts:

midpoints <- c(15)

counts <- c(1000)

Expect equal call:

expect\_equal(midpoints,censor\_df$mids)

expect\_equal(counts, censor\_df$counts)

This test passed, indicating that all datapoints in the same bin works as expected.

The final test was to test with a large dataset.

data <- seq(0.00001, 100, 0.00001)

Predicted output:

midpoints <- seq(5,75,10)

counts <- c(1000000,1000000,1000000,1000000,1000000,1000000,1000000,1000000)

Here we expected to have a million counts.

This output was produced by testthat:

**test\_Armitage.R:45: failure: Test with a large amount of data**

`counts` not equal to censor\_df$counts.

1/8 mismatches

[8] 1e+06 - 999999 == 1

Indicating that this test failed. Not entirely sure why this test failed. My speculation would be that it has to do with numerical approximations, possible issues with rounding? Alternatively, datatype inconsistencies (long int, long float). Perhaps R is having trouble finding a type big enough for this.

# Conclusions : WHAT DID WE LEARN?

* You must save the file before you try to write the tests. Otherwise the tests will run on an out of date tester!
* When passing in a directory to the test\_directory() function, use single quotes. Double quotes can cause issues.
* You must create a test directory.
* You can use testfile() or testdir()
* You have to use the test\_that() function. Inside of this function you write the specific test that you want.

# CONJECTURES & FUTURE QUESTIONS: WHAT COMES NEXT?

* Why didn’t our final test work? Conjecture: Issues with rounding, datatype inconsistencies.

# DOCUMENTATION: WHERE CAN WE SEE THE RESULTS?

https://journal.r-project.org/archive/2011-1/RJournal\_2011-1\_Wickham.pdf