Writing and testing R functions

Hunyong Cho
Department of Biostatistics

Let’s write some functions

Writing an R function - exercise 1

Homework example, but slightly different.

```
set.seed(1)
dat <- matrix(rpois(24, rep(1:4, each = 6)), 6)
```

Goal: writing a function that transforms a data matrix into a compositional data
(Note composition can be made based on row proportion, or column proportion.)

something like this: `compositional(data, margin)`

1. (without writing a function) write a code first.

2. Put it in a function. Replace the values that are potentially changeable with new variables.

Writing an R function - exercise

```
library(tidyverse)
dat2 <- read_tsv("GSE92332_AtlasFullLength_TPM.txt")
head(dat2)
```

Goal: writing a function that returns a scatter plot given two row indices.

something like this: `scatter(gene1, gene2, data)`

1. (without considering a function) Make a scatter plot of 2nd row v.s. 1st row.

2. Put it in a function. Replace the values that are potentially changeable with new variables.

```
# Data source:
https://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE92332&format=file&file=GSE92332%5FAtlasFullLength%5FTPM%2Etxt%2Egz
```

Writing an R function - exercise, continued

Goal2: Using the previous function,
write a function that generates a pdf file including a series of scatter plots.
something like this: `scatterPDF(filename, range1 = 11:20, range2 = 21:30, data)`

1. (without writing a function) Write a for loop that generates scatter plots.

2. Using `pdf()` and `dev.off()`, save the scatter plots into a pdf file.

3. Put it in a function. Replace the values that are potentially changeable with new variables.

Let’s test the functions

Flowchart of testthat package

Let’s test them all at once

test_that (
test_dir (

Filters any unexpected result.

expect_equal()
expect_true()
expect_false()
expect_identical()

expect_output()
expect_message()
expect_error()
expect_warning()
expect_is()
...

)
)

Aside: creating your own expect_*

https://github.com/r-lib/testthat/blob/master/vignettes/custom-expectation.Rmd

```
expect_length <- function(object, n) {
  # 1. Capture object and label
  act <- quasi_label(rlang::enquo(object))

  # 2. Call expect()
  act$n <- length(act$val)
  expect(
    act$n == n,
    sprintf("%s has length %i, not length %i.", act$lab, act$n, n)
  )

  # 3. Invisibly return the value
  invisible(act$val)
}
```

Let’s save the functions into a new file

“F01.functions.R”

And create a new file for testing.

“C01.test.R”

```
library(tidyverse)
library(testthat)
source("F01.functions.R")

set.seed(1)
dat <- matrix(rpois(24, rep(1:4, each = 6)), 6)
dat2 <- read_tsv("GSE92332_AtlasFullLength_TPM.txt")
```

Let’s play with expect_

```
library(testthat)

expect_equal(abs(-5), 5)
expect_equal(abs(-5), -5)                                   -> red flag

expect_is(scatter(1, 2, dat2), "ggplot")
expect_is(scatter(1, 2, dat2), "data.frame")                -> red flag
expect_is(compositional(dat, 1), "matrix")

expect_error(compositional(dat, 1))
```

Wait. Why do we need test_that function?

If you want to test multiple things without test_that function,

you should manually do it.

You cannot have multiple errors in a `for` loop. It will just stop at a first place.

But `test_that` will evaluate everything without stopping, still showing errors!

Let's test. `test_that`

Exercise: Test simultaneously if

- the `scatter` function returns a `ggplot` object for i = 1, j = 2,
- the `scatter` function returns error, if we put unknow object `dat3`, and
- the `scatterPDF` function returns error, if we put two vectors with different lengths.

Exercise: Test if

- the `compositional` function returns a matrix,
- the `compositional` function returns the same dimension of the original matrix,
- the `compositional` function returns a matrix with values ranging [0, 1], and
- the first element of the `compositional` function output is correctly calculated.

No news is good news!

Let's test. `test_dir`

Instead of writing a single code within a `test_that` function,
we can put the components in R files,
put those R files in a folder, and
test them by calling `test_dir`.

1. Make a folder called "test"
2. Open a new R file called "test-scatter.R" and put all the scatter-related expect functions
3. Make "test-compositional.R" similarly.
4. Come back to the "C01.test.R" file and write

Aside: `try()`, `tryCatch()`

When you run a code looping over many values,

there are times you get to have occasional errors but you want to just ignore.

In that case you can use `try()` or `tryCatch()`.