Lab 6: R Functions

Lindsey China (A17023629)

Table of contents

Functions are how we get work done in R. We call functinos to do everything from reading data to doing analysis and outputting plots and results.

All functions have at least three things:

- a **name** (you get to pick this)
- input **arguments** (there can be only one of loads - again your call)
- the **body** (where the work gets done, this code goes between {})

A first silly function

Let's write a function to add some numbers. We can call it `add()`

```
x <- 10
y <- 10
x+y
```

```
[1] 20
```

```
# This works, but it can be changed into a function:

add <- function(x){
  y <- 10
  x+y
}
```

Can I just use my new function? Need to run the original function code to add it to R before being able to use it.

```
add(1)
```

```
[1] 11
```

Let's make it a bit more flexible:

```
add <- function(x, y){
  x+y
}

add(10,1)
```

```
[1] 11
```

```
# or can be written like:

add(x=10,y=1)
```

```
[1] 11
```

If we do:

```
# add(10)
```

Results in an error, there is no argument for y. We can change the function code to make this work:

```
add <- function(x, y=1){
  x+y
}

add(10)
```

```
[1] 11
```

If y is defined in the function code, the value can be overwritten by assigning a new value in the code:

```
add(10,100)
```

```
[1] 110
```

Creating the function `grade()`

Creating the vectors for each student's grades:

```
student1 <- c(100, 100, 100, 100, 100, 100, 100, 90)
student2 <- c(100, NA, 90, 90, 90, 90, 97, 80)
student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
```

The goal is to determine overall grade of each student dropping the lowest score.

```
# Calculate average:

mean(student1)
```

```
[1] 98.75
```

```
# This won't work for student 2 due to the NA in the list, but we can alter the `mean()` c

mean(student2, na.rm=TRUE)
```

```
[1] 91
```

```
# But this isn't helpful for student 3

mean(student3, na.rm=TRUE)
```

```
[1] 90
```

Ok let's try to work with student1 and find/drop the lowest score. Google says to use `min()` and `max()`:

```
min(student1)
```

```
[1] 90
```

This isn't very helpful, using `?min` we find `which.min()`

```r
which.min(student1)
```

```
[1] 8
```

```r
student1[8]
```

```
[1] 90
```

```r
# Or:

student1[which.min(student1)]
```

```
[1] 90
```

How do you use this to exclude the lowest value from the grade calculation? Add a - before the `which.min()`

```r
student1[-which.min(student1)]
```

```
[1] 100 100 100 100 100 100 100
```

```r
mean(student1[-which.min(student1)])
```

```
[1] 100
```

```r
# Can replace student1 with x
x <- student1
mean(x[-which.min(x)])
```

```
[1] 100
```

This won't work for students 2 or 3 though. Our approach to the NA problem: we can replace the NA values with 0's.

First we find the NA values (where they are in the vector)

```
x <- student2
is.na(x)
```

```
[1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

`is.na` lets us find the values that are NA, now I want to make them equal to 0 by overwriting/masking them.

```
x[is.na(x)] <- 0
x
```

```
[1] 100   0  90  90  90  90  97  80
```

Combine `is.na(x)` with making these elements equal to 0, then take this masked vector, drop the lowest and take the mean:

```
x <- student3
x[is.na(x)] <- 0
mean(x[-which.min(x)])
```

```
[1] 12.85714
```

Piece that together into the function `grade()`:

```
grade <- function(x){
  # Mask NA values as 0
    x[is.na(x)] <- 0
  # Drop the lowest assignment grade and get average
  mean(x[-which.min(x)])
}
```

Working function that can be used for each student:

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

Working with the student_homework dataset

```
url <- "https://tinyurl.com/gradeinput"
gradebook <- read.csv(url, row.names=1)
head(gradebook)
```

```
          hw1 hw2 hw3 hw4 hw5
student-1 100  73 100  88  79
student-2  85  64  78  89  78
student-3  83  69  77 100  77
student-4  88  NA  73 100  76
student-5  88 100  75  86  79
student-6  89  78 100  89  77
```

Using the `apply()` function

```
# apply(x, margin, fun, simplify)
# x = the data set or array
# margin = if you'd like to apply function to rows (1) or columns (2)
# simplify = logical, whether or not results should be simplified

apply(gradebook, 1, grade)
```

```
 student-1  student-2  student-3  student-4  student-5  student-6  student-7
     91.75      82.50      84.25      84.25      88.25      89.00      94.00
 student-8  student-9 student-10 student-11 student-12 student-13 student-14
     93.75      87.75      79.00      86.00      91.75      92.25      87.75
student-15 student-16 student-17 student-18 student-19 student-20
     78.75      89.50      88.00      94.50      82.75      82.75
```

Question 2: Identify the top scoring student overall:

```
avg_grades <- apply(gradebook, 1, grade)
which.max(avg_grades)
```

```
student-18
        18
```

Answer: Student 18 has the highest overall grade

Question 3: Which homework assignment was toughest on students:

```r
which.min(apply(gradebook, 2, mean, na.rm=TRUE))
```

```
hw3
  3
```

Answer: Homework 3 was the hardest for students

Question 4: Which assignment was most representative of the overall grade:

```r
# Use correlation function for specific row and average grade values
cor(gradebook$hw1, avg_grades)
```

```
[1] 0.4250204
```

Gives NA for assignments that have NA values

```r
cor(gradebook$hw5, avg_grades)
```

```
[1] NA
```

Mask the NA's to 0:

```r
mask <- gradebook
mask[is.na(mask)] <- 0
```

```r
cor(mask$hw5, avg_grades)
```

```
[1] 0.6325982
```

Use `apply()` to find correlation for all assignments at once:

```r
apply(mask, y=avg_grades, 2, cor)
```

```
      hw1       hw2       hw3       hw4       hw5 
0.4250204 0.1767780 0.3042561 0.3810884 0.6325982 
```

Answer: Homework 5 is most predictive of overall score, homework 2 is the least