# Class 6: R Functions

Scott MacLeod (PID:A16246401)

2024-01-25

## Section 1: Simple R Functions

**Functions** are how we get stuff done! We call functions to do everything useful in R.

One cool thing about R is that it makes writing your own functions comparatively easier.

All functions in R have at least three things:

- a **name** (we get to decide the name)
- one or more **input arguments** (the input to our function)
- the **body** (lines of code that do the work)

```
anyname <- function() {
 # The body with R code
}
```

Let's write a *silly* first function! By silly, we are going to write a simple function for fun!

```
x <- 5
y <- 1
x + y
```

```
[1] 6
```

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

```
addme(69,420)
```

```
[1] 489
```

```r
addme(10)
```

```
[1] 11
```

## Section 2: Lab for Today (01-25-24)

Write a function to grade an entire class! It is best to start with a simplfied version of the problem:

```r
# Example input vectors to start with
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)
```

First, we are just going to find the mean:

```r
mean(student1)
```

```
[1] 98.75
```

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

```
[1] 91
```

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

```
[1] 90
```

This is not fair! There is no way `student3` should have a mean of 90! We need to assign NA a value of 0!

Come back to this NA problem, but things worked for `student1`

We want to drop the lowest score before getting the `mean()`

How do I find the lowest (minimum) score?

```r
student1
```

```
[1] 100 100 100 100 100 100 100  90
```

```r
min(student1)
```

```
[1] 90
```

We know what the score is! But we want to remove it!

I found the`which.min()` function. Which has the possibility of being more useful!

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

```
[1] 8
```

`which.min()` prints 8. Which means that the 8th value in the vector is the lowest number. Now how can I remove this?

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

```
[1] 90
```

We can use the minus trick for indexing

```r
z <- 1:5
z[-3]
```

```
[1] 1 2 4 5
```

Let's remove it!

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

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

This is am ore verbose way to remove it! This helps when looking back at the code.

```r
#Find the lowest score
ind1 <- which.min(student1)
#Remove lowest score and find the mean
mean(student1[-ind1])
```

```
[1] 100
```

Let's take the mean with the removed number!

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

```
[1] 100
```

Wow! Good job for **student1**! They had an average of 100 after we dropped their lowest score, which was 90!

Use a common shortcut and use **x** as my input.

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

```
[1] 100
```

If we tried this with **student2**, we still get the value of "NA".

```r
x <- student2
mean(x[-which.min(x)])
```

```
[1] NA
```

We still have the problem of missing values (aka "NA"). Let's fix this problem!

One idea is to replace NA values with 0!

```r
y <- 1:5
y[y == 3] <- 1000
y
```

```
[1]    1    2 1000    4    5
```

Bummer, this is no good :/

```r
y <- c(1, 2, NA, 4, 5)
y == NA
```

```
[1] NA NA NA NA NA
```

We found the function **is.na()** which will help us identify NA

```r
y
```

```
[1]  1  2 NA  4  5
```

```r
is.na(y)
```

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

Now, how can I remove the NA elements from the vector?

The exclamation mark "flips it"

```r
!c(F,F,F)
```

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

```r
#y[is.na(y)]
```

```r
y[!is.na(y)] <- 1000
y
```

```
[1] 1000 1000   NA 1000 1000
```

Okay, let's put this whole thing together!

```r
x <- student1
#Change NA Values to Zero
x[is.na(x)] <- 0
#Find and remove min value and get mean
```

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

```
[1] 100
```

Let's try **student3**!

```r
x <- student3
#Change NA Values to Zero
x[is.na(x)] <- 0
#Find and remove min value and get mean
mean(x[-which.min(x)])
```

```
[1] 12.85714
```

Last step, now that I have my working code snippet is to make my **grade()** function.

```r
grade <- function(x) {
  #Change NA Values to Zero
x[is.na(x)] <- 0
  #Find and remove min value and get mean
mean(x[-which.min(x)])
}
```

Let's try it!

```r
grade(student1)
```

```
[1] 100
```

```r
grade(student2)
```

```
[1] 91
```

```r
grade(student3)
```

```
[1] 12.85714
```

YAY!

Now let's do the online grade book!

**Question 1** Write a function grade() to determine an overall grade from a vector of student homework assignment scores dropping the lowest single score. If a student misses a homework (i.e. has an NA value) this can be used as a score to be potentially dropped. Your final function should be adquately explained with code comments and be able to work on an example class gradebook such as this one in CSV format: "https://tinyurl.com/gradeinput" [3pts]

```r
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
```

```r
results <- apply(gradebook, 1, grade)
results
```

```
 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** Using your grade() function and the supplied gradebook, who is the top scoring student overall in the gradebook? [3pts]

```r
which.max(results)
```

```
student-18 
        18 
```

**Question 3**From your analysis of the gradebook, which homework was toughest on students (i.e. obtained the lowest scores overall? [2pts]

There are two difference ways to cacluate the worst homework. The first one uses the mean which is not the best way, because it is affected by outliers.

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

```
hw3
  3
```

Using the sum method, we get a different result. It really just depends on what you want to use.

```
which.min(apply(gradebook, 2, sum, na.rm=T))
```

```
hw2
  2
```

**Question 4** Optional Extension: From your analysis of the gradebook, which homework was most predictive of overall score (i.e. highest correlation with average grade score)? [1pt]

The missing homeworks are now 0's!

```
  #Make all (or mask) NA to zero
mask <- gradebook
mask[is.na(mask)] <-0
#mask
```

We can use the `cor()` function for correlation analysis.

```
cor(mask$hw5, results)
```

```
[1] 0.6325982
```

```
cor(mask$hw3, results)
```

```
[1] 0.3042561
```

How do we do it for all of them? I need to use the `apply()` function to run this analysis over the whole course (i.e. masked gradebook).

```r
apply(mask, 2 ,cor, results)
```

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

**Question 5** Make sure you save your Quarto document and can click the “Render” (or Rmark-down”Knit”) button to generate a PDF format report without errors. Finally, submit your PDF to gradescope. [1pt]