Class_06

Kristiana Wong A16281367

R Functions

All functions in R have at least three things:

- A **name** (we get to pick this)
- One or more **input arguments** (the input to our function)
- The **body** (lines of code that do the work)

```
funname <- function(input1, input2) {
  #The body with R code
}
```

Let's write a silly first function to add two numbers:

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

```
[1] 6
```

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

```
addme(1, 1)
```

```
[1] 2
```

```
addme(10)
```

```
[1] 11
```

Lab for Today

Background

In this session you will work through the process of developing your own function for calculating average grades for fictional students in a fictional class. The process will involve starting slowly with small defined input vectors (where you know what the answer should be). Then building up to work with more complex input vectors (with multiple missing elements). Finally, you will turn your code into a function and apply it to a realistic gradebook available online to answer a set of common grading questions. Useful hints are provided in the next section.

Hints

Once you have a working function for vector inputs (such as the student1, student2, and student3 vectors below) you can use the apply() function to work with data frame inputs such as those obtained from read.csv(). Additional functions you will want to explore include mean(), is.na(), which.min(), which.max(), sum(), and cor(). Remember, you can ask for help on any function by typing a question mark before the function name e.g. ?sum. We will walk through many of these steps together in class and in the video review screen-cast. However, attempting on your own before then is highly recommended and will be a big help for following our in class discussion and the the screen-cast review video. As always, if you have questions please ask in person or on Piazza. These are important skills and we want to support you in attaining them as best we can!

```
# 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)
```

Let's just find the average

```
mean(student1)
```

```
[1] 98.75
```

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

```
[1] 91
```

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

```
[1] 90
```

This isn't fair, there is no way Student 3 should have a mean of 90!

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?

```
min(student1, na.rm = FALSE)
```

```
[1] 90
```

```
min(student2, na.rm = FALSE)
```

```
[1] NA
```

```
min(student3, na.rm = FALSE)
```

```
[1] NA
```

I found the 'which.min()' function. Maybe this will be more helpful?

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

```
[1] 8
```

```
which.min(student2)
```

```
[1] 8
```

```
which.min(student3)
```

```
[1] 1
```

Alternatively:

```
#Find the lowest score
student1[which.min(student1)]
```

```
[1] 90
```

```
student2[which.min(student2)]
```

```
[1] 80
```

```
student3[which.min(student3)]
```

```
[1] 90
```

Cool! The output returns to us which element in the vector is the lowest score. Can I remove this value?

Here's a neat little trick:

```
x <- 1:5
x[-3]
```

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

Now to apply this to identify and drop the lowest score and then calculate the mean:

```
#Remove lowest score and find the mean
mean(student1[-8])
```

```
[1] 100
```

```
mean(student2[-8])
```

```
[1] NA
```

```
mean(student3[-1])
```

```
[1] NA
```

We're still getting an NA error though. Hm….

Firstly though, let's do a common shortcut to condense the work:

```
x <- student1
y <- student2
z <- student3
```

One suggestion was to convert the NA values to zero. Lets first see which values are NA within each vector:

```
is.na(x)
```

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

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

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

```
is.na(z)
```

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

How can I remove the NA elements from the vector?

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

Alright. Lets solve this:

```
x <- student1

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

```
[1] 100
```

```
y <- student2

#Change NA values to Zero
y[is.na(y)] <- 0
#Find and remove min value and get the mean
mean(y[-which.min(y)])
```

```
[1] 91
```

```
z <- student3

#Change NA values to Zero
z[is.na(z)] <- 0
#Find and remove min value and get the mean
mean(z[-which.min(z)])
```

```
[1] 12.85714
```

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

```
grade <- function(student1, student2, student3) {

   x <- student1

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

Lets see if it worked:

```
grade (student1)
```

```
[1] 100
```

```
grade (student2)
```

```
[1] 91
```

```
grade (student3)
```

```
[1] 12.85714
```

Yay! It worked!

Now read the online gradebook (CSV file)

```
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
```

Q1. 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"

```
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
```

Q2. Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook?

```
  max(results)
```

```
[1] 94.5
```

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

```
student-18
        18
```

Q3. From your analysis of the gradebook, which homework was toughest on students (i.e. obtained the lowest scores overall?

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

```
hw2
  2
```

Q4. Optional Extension: From your analysis of the gradebook, which homework was most predictive of overall score (i.e. highest correlation with average grade score)?

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

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

```
  cor(gradebook$hw1, results)
```

```
[1] 0.4250204
```

We need to use the 'apply()' function to apply this over the whole gradebook

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

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

Homework 5 was most predicitive.

Q5. Make sure you save your Quarto document and can click the “Render” (or Rmarkdown”Knit”) button to generate a PDF foramt report without errors. Finally, submit your PDF to gradescope.