Class 6: R Functions

Justin Lu (PID: A16318305)

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

At least 3 things in all functions in R:

1. A **name** (you get to pick this)
2. Input **arguments** (there can only be one or loads - again your call)
3. The **body** (where the work gets done, the code between the brackets)

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
```

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

Can I just use my new function?

```
add (1)
```

```
[1] 11
```

Let's make it a bit more flexible

```
add <- function (x,y=10) {
  x + y
}
 add(10,5)
```

```
[1] 15
```

```
add(1)
```

```
[1] 11
```

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

```
[1] 110
```

2nd example grade() function

Write a function to grade student work.

We will start with a simple version of the problem and the following example student vectors

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)

min(student1)
```

```
[1] 90
```

```
mean(student1)
```

```
[1] 98.75
```

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

```
[1] 91
```

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

```
[1] 90
```

OK lets try to work with student1 and find and drop the lowest score

```
  student1
```

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

Google told me about min() and max()

```
  min(student1)
```

```
[1] 90
```

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

```
[1] 8
```

```
  student1[8]
```

```
[1] 90
```

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

```
[1] 90
```

```
  student1[-8]
```

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

```
mean(student1)
```

```
[1] 98.75
```

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

```
[1] 100
```

1st task is to find NA values (where they are in the vectors)

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

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

I have found the NA (true values) from `is NA()` now I want to make them equal to zero (overwrite/mask them etc.)

I want to combine the `is na(x)` with making these elements equal to zero. Then take this "masked" (vector of student scores with NA values as 0) and drop the lowest and get the mean.

```
x <- student2
x[is.na(x)] <- 0
x
```

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

```
## average without the dropped score
mean(x)
```

```
[1] 79.625
```

```
## average with the dropped score
mean(x[-which.min(x)])
```

```
[1] 91
```

```
y <- student3
y[is.na(y)] <- 0
y
```

```
[1] 90  0  0  0  0  0  0  0
```

```
## average without the dropped score
mean(y)
```

```
[1] 11.25
```

```
##average with the dropped score
mean(y[-which.min(y)])
```

```
[1] 12.85714
```

Now I can turn my awesome snipet into my first function

```
grade <- function (x)
{
  x[is.na(x)] <- 0
  mean(x[-which.min(x)])
}
```

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

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

```
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
```

The `apply()` function in R is super useful but can be a little confusing to begin with. Lets have a look at how it works

```
#Getting the mean of each student with the lowest score dropped by applying grade to grade
ans <- apply(gradebook, 1, grade)
ans
```

```
 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

```
#We can get find the location of the highest value using the which.max function, using onl

which.max(ans)
```

```
student-18
        18
```

```
max(ans)
```

```
[1] 94.5
```

The code ran that Student 18 had the top scoring student with a score of 94.5

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

```
#This would give us hardest homework if we removed all the NA values because of extenuatin

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

```
[1] 80.8
```

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

```
hw3
  3
```

```
#We can use the mean function to get the average of each homework assignment across all st

gradebook[is.na(gradebook)] <- 0

#Now we can get the mean
min(apply (gradebook, 2, mean))
```

```
[1] 72.8
```

```
which.min(apply(gradebook,2, mean))
```

```
hw2
  2
```

HW 2 had the lowest average with an average score of 72.80, so HW2 was the toughest for students overall if we set all NA scores to 0. However, HW3 was the hardest with an average score of 80.8 if we remove all the NAs from the dataset.

> 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)?

```
#correlation function, closer to 1 has higher correlation
cor(gradebook$hw1, ans)
```

```
[1] 0.4250204
```

```
mask <- gradebook
mask[is.na(mask)] <- 0
cor(mask$hw5, ans)
```

```
[1] 0.6325982
```

I want to apply this function to all homeworks by using the `apply()` to examine the correlation of each assignment in the masked gradebook to the overall score for each student in the class.

```
#error comes from using a regular apply function because cor requires 2 arguments, so an e

apply(mask, 2, cor, y = ans)
```

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

From the results, HW2 has the lowest correlation value and HW5 has the highest correlation value meaning that HW5 is the best predictor of overall score compared to other homeworks.