Class 06: R Functions

Helen Le (PID: 16300695)

All about functions in R

Functions are the way we get stuff done in R. We call a function to read data, compute stuff, plot stuff, etc.

R makes writing functions accessible but we should always start by trying to get a working snippet of code first before we write our function.

Today's lab

We will grade a whole class of student assignments. We will always try to start with a simplified version of the problem.

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 adequately 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]

```
# 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)

grade1 <- mean(student1)
grade1
```

```
[1] 98.75
```

If we want the average, we can use the `mean()` function.

I can use the `min()` function to find the lowest value.

```
min(student1)
```

```
[1] 90
```

I found the `which.min()` function that may be useful here. Let's try:

```
student1
```

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

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

```
[1] 8
```

```
student1[8]
```

```
[1] 90
```

```
# which is the same as...
student1[which.min(student1)]
```

```
[1] 90
```

```
# however, that's not what we want-- we want everything BUT the min value
# instead, we could do:
student1[-8] # hard coding
```

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

```
# which is the same as...
student1[-which.min(student1)]
```

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

I can use the minus syntax trick to get everything but the element with the min value.

I have my first working snippet of code :]

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

```
[1] 100
```

Let's test this code on student 2:

```
student2
```

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

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

```
[1] NA
```

Where is the problem?

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

```
[1] 91
```

Typing `?mean` into the RConsole, we see that the problem lies with the `mean()` with NA input. It returns NA by default but we can change this so that NA values are stripped from the list.

How about student 3?

```
student3
```

```
[1] 90 NA NA NA NA NA NA NA
```

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

```
[1] 90
```

However, this is unfair since they missed many assignments but judging from their mean grades, it seems like they did great.

I want to stop working with `student1`, `student2`, etc. and typing it out every time so instead, let's work with an input called `x`.

```
x <- student2
x
```

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

We want to override the NA values with zero – if you miss a homework, you score 0 on this homework.

Google and Claude told me about the `is.na()` function. Let's see how it works.

```
x
```

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

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

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

```
x[is.na(x)]
```

```
[1] NA
```

We can use logicals to index a vector.

```
y <- 1:5
y
```

```
[1] 1 2 3 4 5
```

```
y > 3
```

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

```
y[y > 3]
```

```
[1] 4 5
```

```
y[y > 3] <- 100
y
```

```
[1]   1   2   3 100 100
```

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

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

Let's combine this with the previous code used with student1: This is my working snippet of code that solves the problem for all my example student inputs :]

```
x <- student3
# Mask NA values to zero
x[is.na(x)] <- 0
# Drop lowest score and get the mean
mean(x[-which.min(x)])
```

```
[1] 12.85714
```

Make this into a function:

```
grade <- function(x) {
  # Mask NA values to zero
  x[is.na(x)] <- 0
  # Drop lowest score and get the mean
  mean(x[-which.min(x)])
```

```
}
```

Use this function:

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

```
gradebook <- read.csv("https://tinyurl.com/gradeinput", row.names=1)
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
student-7   89 100  74  87 100
student-8   89 100  76  86 100
student-9   86 100  77  88  77
student-10  89  72  79  NA  76
student-11  82  66  78  84 100
student-12 100  70  75  92 100
student-13  89 100  76 100  80
student-14  85 100  77  89  76
student-15  85  65  76  89  NA
student-16  92 100  74  89  77
student-17  88  63 100  86  78
```

```
student-18  91  NA 100  87 100
student-19  91  68  75  86  79
student-20  91  68  76  88  76
```

Figuring out the `apply()` function:

```
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
```

Typing `?apply()` into the RConsole, we see that the function takes in the arguments x, MARGIN, and FUN. - We input the list of vectors (gradebook) into x. - We want the function to be applied over rows, which is done with MARGIN=1 - We want grade() to be the function used, which is done with FUN=grade

Q2. Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook? [3pts]

```
top_scoring <- which.max(ans)
ans[top_scoring]
```

```
student-18
      94.5
```

The top scoring student is student 18 with an average of 94.5.

Q3. From your analysis of the gradebook, which homework was toughest on students (i.e. obtained the lowest scores overall?) [2pts]

```
lowest_hw <- apply(gradebook, 2, grade)
lowest_hw
```

```
     hw1      hw2      hw3      hw4      hw5
89.36842 76.63158 81.21053 89.63158 83.42105
```

```
which.min(lowest_hw)
```

```
hw2
  2
```

Homework 2 was the toughest on students and obtained the lowest scores overall.

Another way to do this is:

```
mask <- gradebook

mask[is.na(mask)] <- 0
hw.ave <- apply(mask, 2, mean)
hw.ave
```

```
  hw1   hw2   hw3   hw4   hw5
89.00 72.80 80.80 85.15 79.25
```

```
which.min(hw.ave)
```

```
hw2
  2
```

The values of hw.ave here are different from the above answer because the NA values– which are zero– aren't removed, making the list of average homework scores lower.

Or:

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

```
     hw1      hw2      hw3      hw4      hw5
89.00000 80.88889 80.80000 89.63158 83.42105
```

We could take the sum:

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

```
 hw1  hw2  hw3  hw4  hw5
1780 1456 1616 1703 1585
```

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)? [1pt]

```
correlation <- apply(mask, 2, cor, y=ans)
correlation
```

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

```
which.max(correlation)
```

```
hw5
  5
```

Homework 5 was the most predictive of the overall score.