Class06: R Functions

Alvin Cheng (A16840171)

#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. etc.

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

Today lab

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

```
# 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)
```

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

```
avg_score = mean(student1)
avg_score
```

```
[1] 98.75
```

Let's be nice instructors and drop the lowest score so the answer here should be 100.

I found the `which.min()` function that may be useful here. How does it work? Let's just try it:

```
student1
```

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

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

```
[1] 8
```

`which.min` gives you the position of the lowest score

```
student1
```

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

```
min(student1)
```

```
[1] 90
```

`min()` gives you the lowest score

Putting a minus side in front should eliminate the lowest score. I can use the minus syntax trick to get everything but the element with the min value.

```
student1[-8]
```

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

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

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

This line of code below will give you the average of student 1 score after dropping the student's lowest score!

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

```
[1] 100
```

Let's test on the other students

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

```
[1] NA
```

where is the problem? Oh it is the the `mean()` with NA input returns NA by default but I can change this...

```
student2
```

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

```
mean(student2) # this produces an error
```

```
[1] NA
```

```
mean(student2, na.rm=T) # this does not
```

```
[1] 91
```

```
student3
```

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

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

```
[1] 1
```

```
min(student3)
```

```
[1] NA
```

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

```
[1] 90
```

No bueno. The student only submitted one of the homework and did not for the other yet the algorithim gave him a 90%! We need to fix this!

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

```
x = student2
x
```

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

We want to overwrite the NA values with zero - if you miss a homework you score zero on this homework.

Bard has told me about the `is.na()` function to find the na vlaues.

```
x
```

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

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

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

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
```

There are multiple ways to replace NA, but one way is showed! (Using bard)

```
x <- replace(student2, is.na(student2), 0)
x # in order not to overwrite student2, I used the variable x
```

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

```
# another way is here
x[is.na(x)] <- 0
x
```

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

Now let's find the average

```
mean(x)
```

```
[1] 79.625
```

Let's do student 3. Keep in mind, there is another similar way to do this

```
student3_drop <- replace(student3, is.na(student3), 0)
student3_drop # replacing all NA with 0
```

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

```
#student3_replaced <- student3_drop[-which.min(student3_replaced)] # dropping the lowest v
student3_replaced <- student3_drop[-which.min(student3_drop)] # dropping the lowest value
avg_student3 <- mean(student3_replaced, na.rm=TRUE) # finding the average now
avg_student3
```

```
[1] 12.85714
```

^^ This is my working snippet of code!

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" [3pts]

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

Use this function: (Remember to run the code above first!)

```
#this should run the function "grade" on the data of student 1-3
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

We need to read the gradebook for an overview of all the students' grades

```
gradebook <- read.csv("https://tinyurl.com/gradeinput", row.names=1) # reads the file
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
```

```
grades <- apply(gradebook,1,grade) # applies the grade function to each row
grades # displays the grades of all the students
```

```
 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? [3pts]

```
# Apply the `grade()` function to each row of the data frame
grade_max <- apply(gradebook, 1, grade)
#grade_max

#an easier way
highest_scoring_student = which.max(grade_max)
highest_scoring_student
```

```
student-18
        18
```

```
# Find the highest score
#highest_score <- max(grade_max)

# Find the student with the highest score
#highest_scoring_student <- names(grade_max)[grade_max == highest_score]

# Print the highest scoring student
#print(highest_scoring_student)
```

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

```
# Apply the `grade()` function to each row of the data frame
grade_min <- apply(gradebook, 2, grade)
grade_min
```

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

```
which.min(grade_min)
```

```
hw2
  2
```

Another way to do it

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

a <- apply(mask,2,mean)
lowest_hw <- which.min(a)
lowest_hw
```

```
hw2
  2
```

```
#which.min(apply(mask,2,mean))
```

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]

```
which.max(apply(mask,2,cor,y=grades))
```

```
hw5
  5
```