Class06_Functions

Krysten Jones (PID: A10553682)

All About Functions

every function in R has at least 3 things - name (you pick) - arguments (the input(s) to your function) - body ()

Today we will write a function to grade a class of student assignment scores (e.g. homework, ect.)

First I will work with a simplified vector input where I know what the answer should be.

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)
```

Lets first get the average for student 1

```
avg_student1 <- mean(student1)
avg_student1
```

```
[1] 98.75
```

How can we drop the lowest score? Use the `min()` function to find the lowest score. You can also ask which (element) location in the vector contains the minimum `which.min(x)`. You can nest the `which.min()` function inside where you would normally call a specific vector using `vector[position]`. To get everything except that location, add a `-` in the argument location. In turn, you can nest that entire function inside the mean function to get the mean for the student without the lowest score.

```
min(student1)
```

```
[1] 90
```

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

```
[1] 8
```

```
student1[8]
```

```
[1] 90
```

```
student1[-8]
```

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

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

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

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

```
[1] 100
```

Note: if you put the minus sign instead outside of the [] you will return the negative of the other value

To remove all the NA scores from a group, you can use the argument `na.rm = TRUE`, by default remove na is set to false (aka don't remove the NA's) `na.rm = FALSE`

```
x <- student2
mean(x[-which.min(x)], na.rm=TRUE)
```

```
[1] 92.83333
```

However this wouldn't work for student 3 who has multiple NAs, so instead we can mask NAs with a 0. The rational being if you don't do the HW you get 0 points. If you don't know how to do this, you can always use an AI (like Claud.io which is a free version of ChatGPT4).

We can use the `is.na()` function to rind where the missing homeworks are in the input vector to get a logical. To get the whole vector, but with the NA replaced by 0, simply next it within the student vector. So in the student vector, if something is NA, in the same vector set it to 0

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

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

Now lets see if we can run it all together for student 3.

```
x <- student3
# Mask NA to 0
x[is.na(x)] <- 0
# find the mean dropping the lowest score, it will only drop 1, if you wanted to drop 2, y
mean(x[-which.min(x)])
```

```
[1] 12.85714
```

This now completes the body of our function, now we need to just add the names and arguments.

```
grade <- function(x) {
  # this is the body of the function below
  # Mask NA to 0
  x[is.na(x)] <- 0
  # find the mean when dropping the lowest score
  mean(x[-which.min(x)])
}
```

DONT FORGET TO RUN YOUR CODE, if you check in the global environment in the right, you'll have a function section with your code name

An alternative way to name your function is to select a code chunk and go to code button above –> extract function

Now to answer question 1, we need to read the gradebook

```
gradebook <- read.csv("https://tinyurl.com/gradeinput")
head(gradebook)
```

```
          X hw1 hw2 hw3 hw4 hw5
1 student-1 100  73 100  88  79
2 student-2  85  64  78  89  78
3 student-3  83  69  77 100  77
4 student-4  88  NA  73 100  76
5 student-5  88 100  75  86  79
6 student-6  89  78 100  89  77
```

Hmmm, it's reading the row names as an additional student column. So lets change it to the row names by changing the arguments

```
gradebook <- read.csv("https://tinyurl.com/gradeinput", 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
```

Much better. Now in other languages we would start to write loops. However, if you're writing loops in R, something is likely wrong. Instead, you'll use the tidyverse package and use `apply()` as a function.

```
Q1_ans <- apply(gradebook,
    # the 1 here means each vector is in a row (e.g. each student). If they were in column
      1,
    # our function name, you don't need the parentheses or quotation marks here
      grade)
Q1_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
```

Question 2. Find who is the top scoring student

```
# this is the maximum grade
max(Q1_ans)
```

```
[1] 94.5
```

```
# this is which row (aka student), has the top grade
which.max(Q1_ans)
```

```
student-18
        18
```

Question 3.

From your analysis of the gradebook, which homework was toughest on students (i.e. obtained the lowest scores overall?) Let's assume that NAs just aren't included.

```
Q3_ans <- apply(gradebook, 2, mean, na.rm = TRUE)
which.min(Q3_ans)
```

```
hw3
  3
```

What if we assume that the NAs actually mean the hw problems were super hard and want to replace it with 9

We want to make sure that we are re-defining if we're changing the original gradebook because otherwise it will change our original data

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

```
           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   0  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   0  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   0
student-16  92 100  74  89  77
student-17  88  63 100  86  78
student-18  91   0 100  87 100
student-19  91  68  75  86  79
student-20  91  68  76  88  76
```

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

```
hw2
  2
```

```
which.min(apply(mask, 2, sum))
```

```
hw2
  2
```

```
which.min(apply(mask, 2, median))
```

```
hw2
  2
```

This is looking at different values outside the mean and see if you still get the same type

Question 4

From your analysis of the gradebook, which homework was most predictive of overall score (i.e. highest correlation with average grade score)? This will tell you which were the best/more indicative assignment. We want your assessments/scores to be indicative of how as student is learning/doing in the course. To do this we're going to use a pearson correlation.`cor()` default is set to pearson, but you can also run others

```
#looking at the hw that people did the worst with (this should be a low correlation)
# remember that Q1_ans is where we stored the student's overall scores
cor(mask$hw2, Q1_ans)
```

```
[1] 0.176778
```

```
# would be better to look at a different hw. We can go through and manually pick all of th
cor(mask$hw5,Q1_ans)
```

```
[1] 0.6325982
```

```
#Lets find out which one students did the best using a median
which.max(apply(mask, 2, median))
```

```
hw1
  1
```

```
cor(mask$hw1,Q1_ans)
```

```
[1] 0.4250204
```

Hmmm, it seems like our random pick had a higher correlation than either the highest or lowest scoring hw assignments. Lets have our program figure out which hw has the highest correlation

```
# if you look at the help page, it shows "..." as possible arguments that you can apply to
Q4_ans <- apply(mask, 2, cor, y=Q1_ans)
#if you just want the max, you can
best_hw_indicator <- which.max(Q4_ans)
```

```
best_hw_indicator
```

```
hw5
  5
```