class06

Adithi Kumar (PID:A16653979)

#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 our function.

##Todays lab

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

```r
# 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 score for one student , we can use the 'mean()' function

```r
mean(student1)
```

```
[1] 98.75
```

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

```r
min(student1)
```

```
[1] 90
```

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

```r
student1
```

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

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

```
[1] 8
```

```r
mean(student1[-8])
```

```
[1] 100
```

'which.min' give us the position of the lowest score. 'vector[which.min(vector)]' gives us the lowest score.

Since we know the 8th position is the lowest score, we can find the mean without 8 by using the minus sign. *(- = without)*

All together this code is the best way to get the overall grade while dropping the lowest score.

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

```
[1] 100
```

Let's test on the other students:

```r
student2
```

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

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

```
[1] 8
```

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

```
[1] NA
```

where is the problem? - oh it is the 'mean()' with NA input returns NA by default… but I can change this

By making na.rm as TRUE, the NAs are stripped before the computation proceeds

```
mean(student2)
```

```
[1] NA
```

```
mean(student2, trim =0, na.rm=TRUE)
```

```
[1] 91
```

Student3 has multiple NAs. the na.rm = TRUE command gets rid of ALL NAs so student3 (who only did one assignment) will get a mean score of 90-- which isn't fair.

```
student3
```

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

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

```
[1] 90
```

I want to stop working with 'student1' , 'student2', and 'student3' and typing it out every time so 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 a zero - if you miss a homework, you score zero on this homework.

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

```r
x
```

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

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

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

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

```
[1] NA
```

We can use logicals to index a vector. vector[] will only put out the TRUE values. You can then assign them new values (overwrite them).

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

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

```r
y<3
```

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

```r
y[y<3]
```

```
[1] 1 2
```

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

```
[1] 100 100   3   4   5
```

This is my working snippet of code that solves the problem for all my example student inputs :-)

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

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

```r
mean(x[-which.min(x)])
```

```
[1] 91
```

This works for student 3 too!

```r
x <- student3
x
```

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

```r
#make NA equal to 0
x[is.na(x)] <- 0
# drop the lowest score and get the mean
mean(x[-which.min(x)])
```

```
[1] 12.85714
```

Now, we need to turn this into a function to answer Q1:

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]

```r
grade <- function (x){
  #make NA equal to 0
  x[is.na(x)] <- 0
```

```
    #drop the 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
```

We need to read the gradebook. By using the ‘apply’ function, we can apply the ‘grade’ function that takes the average while dropping the lowest score to the gradebook.

TO use apply, Apply(whatever you want the function to be applied to, Margin, function)

Margin determines whether the function is applied to the row (Margin =1), the column (margin =2), or both (c(1,2))

```r
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 overall in the gradebook?

```r
which.max(ans)
```

```
student-18 
        18 
```

```r
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
```

Student 18 was the top scoring student overall in the gradebook.

Q3:

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

```r
mean_homework_scores <- apply(gradebook, 2, mean, na.rm =T)
which.min(mean_homework_scores)
```

```
hw3
  3
```

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

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

The homework that was toughest on the students was homework 2.

Q4:

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

I used Bard (Google's AI) to try to find a function that would help. It gave me the function 'corr()' which would give a correlation coefficient; the larger the correlation coefficient, the more predictive the homework was on the overall score.

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


cor_hwscores <- apply(gradebook, 2, cor, y=ans)

which.max(cor_hwscores)
```

```
hw5
  5
```

Homework 5 has the highest correlation with average grade score.