R Functions Lab (Class 06)

Mason Lew (PID: A17533139)

> 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"

First I will input the data set and visualize the beginning of the data to see what I am working with.

```
url <- "https://tinyurl.com/gradeinput"

gradebook <- read.csv(url)

head(gradebook,10)
```

```
            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
7   student-7  89 100  74  87 100
8   student-8  89 100  76  86 100
9   student-9  86 100  77  88  77
10 student-10  89  72  79  NA  76
```

Now I will write a function that will determine the mean of the grades per row. The first problem I run into is that the first row is a character and not a numeric number. For now, to sidestep this issue, I will use `data[-1]` which will read the data starting at the second column.

```
grade <- function(data) {
  Mean <- rowMeans(data[,-1])
  Mean
}
grade(gradebook)
```

```
 [1] 88.0 78.8 81.2   NA 85.6 86.6 90.0 90.2 85.6   NA 82.0 87.4 89.0 85.4   NA
[16] 86.4 83.0   NA 79.8 79.8
```

Currently, the function cannot take in the `NA` values. I will now try to change this using the the `is.na()` function call.

```
grade <- function(data) {
  data[is.na(data)] <- 0
  Mean <- rowMeans(data[,-1])
  Mean
}
grade(gradebook)
```

```
 [1] 88.0 78.8 81.2 67.4 85.6 86.6 90.0 90.2 85.6 63.2 82.0 87.4 89.0 85.4 63.0
[16] 86.4 83.0 75.6 79.8 79.8
```

Great now that I have achieved the mean, I can work on dropping the lowest score from each row. To do this I will use the `apply()` function

```
grade <- function(data) {
  data[is.na(data)] <- 0
  data2 <- data[-apply(data[,-1], 1, min)]
  Mean <- rowMeans(data2[,-1])
  Mean
}
grade(gradebook)
```

```
 [1] 88.0 78.8 81.2 67.4 85.6 86.6 90.0 90.2 85.6 63.2 82.0 87.4 89.0 85.4 63.0
[16] 86.4 83.0 75.6 79.8 79.8
```

OH NO... When I tried the solution above, it turns out I was not able to drop the lowest score with this method due to my input being a dataframe still. To remedy this, I changed the way I was going to *drop* the lowest score. Now, I will simply set the lowest score in the each row to

0 using the `zero` function I wrote in combination with `apply()`. To ensure Now because the lowest score is 0, it will not affect the summing of a row. everything remains as a dataframe, I used `as.data.frame()` and `t()`. To ensure everything remained numeric (outside of column 1) I used `as.numeric`. Finally I manually computed the mean scores of summing each row and dividing by the amount of columns -2 (-1 for the student name column, and -1 to account for the *dropped score*).

```
grade <- function(data) {
  data[is.na(data)] <- 0
  data

  zero <- function(Row) {
    Row[which.min(Row)] <- 0
    Row
  }

  data2 <- as.data.frame(t(apply(data, 1, zero)))
  colnames(data2) <- colnames(data)
  data2[,-1] <- lapply(data2[,-1], as.numeric)
  data2
  Sum <- rowSums(data2[,-1])
  Sum
  Final_grade = Sum/(ncol(data2) - 2)
  Final_grade
}
grade(gradebook)
```

```
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
```

```
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
```

```
 [1] 91.75 82.50 84.25 84.25 88.25 89.00 94.00 93.75 87.75 79.00 86.00 91.75
[13] 92.25 87.75 78.75 89.50 88.00 94.50 82.75 82.75
```

Now, I must reassemble the dataframe so that each student is labeled with their score. To do this I will use `data.frame`

```r
#creates the function with in the format `function_name <- function(`input_arguments`) {'body
grade <- function(data) {

  #Sets all data = to NA to 0 so it can be accounted for in the grade
  data[is.na(data)] <- 0

  #Creates new function that will turn the lowest value of any imputed vector to 0
  zero <- function(Row) {
    Row[which.min(Row)] <- 0
    Row
  }

  #Keeps output as a dataframe while apply the 'zero' function to each row
  data2 <- as.data.frame(t(apply(data, 1, zero)))

  #keeps all values outside the first column as numeric values
  data2[,-1] <- lapply(data2[,-1], as.numeric)

  #Sums reach row excluding the first column
  Sum <- rowSums(data2[,-1])

  #computes the mean manually by dividing the sum of each row by the number of columns - 2
  Final_grade = Sum/(ncol(data2) - 2)

  #The student numbers are collected in a vector
  student <- gradebook[,1]

  #data frame is reconstructed with student numbers in the first column and grades in the se
  complete_df <- data.frame(Student = student, Grade = Final_grade)
```

```r
  #final dataset is printed out for presentation
  complete_df
}

Complete_df <- grade(gradebook)
```

```
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
Warning in which.min(Row): NAs introduced by coercion
```

```r
Complete_df
```

```
      Student Grade
1   student-1 91.75
2   student-2 82.50
3   student-3 84.25
4   student-4 84.25
5   student-5 88.25
6   student-6 89.00
7   student-7 94.00
8   student-8 93.75
9   student-9 87.75
10 student-10 79.00
11 student-11 86.00
```

```
12 student-12 91.75
13 student-13 92.25
14 student-14 87.75
15 student-15 78.75
16 student-16 89.50
17 student-17 88.00
18 student-18 94.50
19 student-19 82.75
20 student-20 82.75
```

> Q2 Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook?

To do this, I will use the `which.max()` function on the second column of the dataframe

```
#Uses final dataframe from Q1 to find the highest grade (column 2) then returns the column 1
Complete_df[which.max(Complete_df[,-1]),1]
```

```
[1] "student-18"
```

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

To do this, I will Sum each Column with `colSums` then compare.

```
#Sums all columns up aside from the first one
column_sums <- colSums(gradebook[,-1])

#Creates a vector with all column names in it
col_names <- colnames(gradebook)

#returns the vector value given by the lowest of the summed columns
col_names[which.min(column_sums)]
```

```
[1] "hw2"
```

> 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)?

Here I will use `cor()` to determine the correlation of separate columns.

```r
#Found the correlation between each hw score and the final grade received
cor1 <- cor(gradebook$hw1, Complete_df$Grade)
cor2 <- cor(gradebook$hw2, Complete_df$Grade)
cor3 <- cor(gradebook$hw3, Complete_df$Grade)
cor4 <- cor(gradebook$hw4, Complete_df$Grade)
cor5 <- cor(gradebook$hw5, Complete_df$Grade)

#returned the column name of the hw that had the highest cor (+1 due to the name column skipp
total_cor <- c(cor1, cor2, cor3, cor4, cor5)
col_names[which.max(total_cor) + 1]
```

```
[1] "hw1"
```