Class 06: R Functions

Lixin Sun

2023-04-21

In this class, we will develop our own **R functions** to calculate average grades in a fictional class.

Simplified Version

We will start with a simplified version of the problem, just calculating the average grade of one student.

```
# 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)
```

First, calculate the average score of the assignments using `mean()`.

```
mean(student1)
```

```
[1] 98.75
```

Then, find the minimum score using `which.min()`.

```
student1
```

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

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

```
[1] 8
```

Average the first 7 assignments.

```
mean(student1[1:7])
```

```
[1] 100
```

Another way to select the first 7 assignments.

```
student1[1:7]
```

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

```
student1[-8]
```

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

We can combine the codes into one line to find the average after dropping the lowest score.

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

```
[1] 100
```

Save the variable of the vector with the lowest score dropped.

```
student1_drop_lowest=student1[-which.min(student1)]
student1_drop_lowest
```

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

```
mean(student1_drop_lowest)
```

```
[1] 100
```

Try applying the code for student2.

```
  student2
```

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

```
  student2_drop_lowest = student2[-which.min(student2)]
  student2_drop_lowest
```

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

The above code is not working because NA is recognized as a character instead of number. There is a way to calculate the mean after removing the missing values.

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

```
[1] 91
```

However, this code doesn't work for students with multiple missing scores, like student3.

```
  student3
```

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

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

```
[1] 90
```

Now, find the position of missing scores without removing them for student2.

```
  student2
```

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

```
  is.na(student2)
```

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

```
which(is.na(student2))
```

```
[1] 2
```

Apply the code for student3 and we can see there are multiple missing values.

```
student3
```

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

```
which(is.na(student3))
```

```
[1] 2 3 4 5 6 7 8
```

Then, mask the missing values with zeros. Apply the code for student3 and find the new mean after dropping the lowest score.

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

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

```
student3[is.na(student3)] <- 0
student3
```

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

```
mean(student3)
```

```
[1] 11.25
```

```
mean(student3[-which.min(student3)])
```

```
[1] 12.85714
```

```
student3_drop_lowest = student3[-which.min(student3)]
mean(student3_drop_lowest)
```

```
[1] 12.85714
```

Now, build a function that combines all of the codes above.

```
# Here's the working code.
x <- c(100,75,50,NA)
x[is.na(x)] <- 0
x_drop_lowest <- x[-which.min(x)]
mean(x_drop_lowest)
```

```
[1] 75
```

Q1. Function grade()

```
#' Calculate the average score for a vector of homework scores,
#' considering NA values as zeros and dropping the lowest score.
#'
#' @param x A numberic vector of homework scores
#'
#' @return Teh average value of homework scores
#' @export
#'
#' @examples
#'
#' student <- c('100', '50', NA)
#' grade(student)
#'
grade <- function(x) {
    # Mask NA values with zeros
    x[is.na(x)] <- 0
    # Drop the lowest score
    x_drop_lowest <- x[-which.min(x)]
    # Calculate the average score
    mean(x_drop_lowest)
}
```

Let's test the function.

```
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)

grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student3)
```

```
[1] 12.85714
```

Now, apply the function to a gradebook from this URL https://tinyurl.com/gradeinput

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

#row.names
gradebook <- read.csv(URL, 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
```

If we directly run the code below, an error message saying "Error in which.min(x) : 'list' object cannot be coerced to type 'double' " will appear. This is because the input of the function we created should be a vector, while `gradebook` is a dataframe.

```
#grade(gradebook)
```

To solve this issue, we need to use the function `apply(array, MARGIN, function)` and running it by **rows** with `MARGIN = 1`.

```
apply(gradebook, 1, grade)
```

```
 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]

Answer: The top scoring student overall is student 18 and the maximum score is 94.5.

```
which.max(apply(gradebook, 1, grade))
```

```
student-18
        18
```

```
max(apply(gradebook, 1, grade))
```

```
[1] 94.5
```

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

Answer: The toughest homework will be hw2 considering the mean and considering missing homework as 0. If we consider the mean without missing homework, the toughest homework will be hw3. If we consider the median without missing homework, then the toughest homework will be hw2.

```
# Replace NA values with zeros.
gradebook[is.na(gradebook)]<- 0
# Apply the function to find averages of each homework.
apply(gradebook, 2, mean)
```

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

```
# Apply the mean function with an arguement that removes NA values.
gradebook <- read.csv(URL, row.names = 1)
apply(gradebook, 2, mean, na.rm = TRUE)
```

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

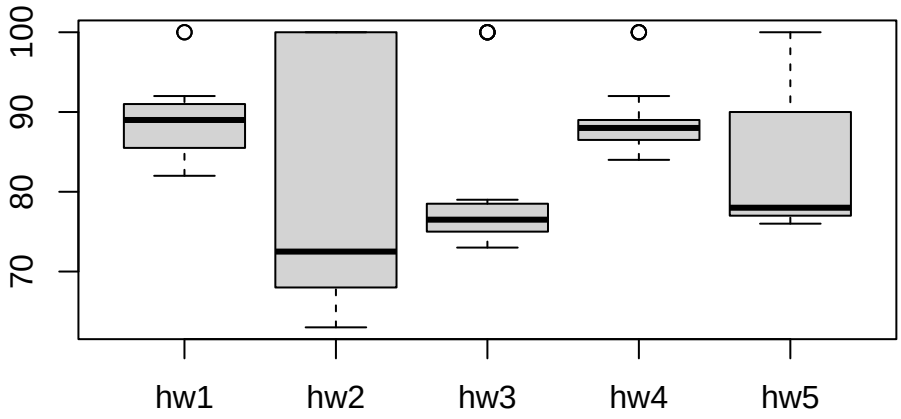
If we use the median instead of the mean as a measure of overall score...

```
apply(gradebook, 2, median, na.rm = TRUE)
```

```
 hw1  hw2  hw3  hw4  hw5
89.0 72.5 76.5 88.0 78.0
```

If we use some plots...

```
boxplot(gradebook)
```



Q4. From your analysis of the gradebook, which homework was most predictive of overall score (i.e. highest correlation with average grade score)? [1pt]

Answer: Hw5 was the most predictive of overall score with the highest correlation of 0.6325982.

```
overall_grades = apply(gradebook, 1, grade)
overall_grades
```

```
 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
```

```
gradebook$hw1
```

```
 [1] 100  85  83  88  88  89  89  89  86  89  82 100  89  85  85  92  88  91  91
[20]  91
```

```r
cor(overall_grades, gradebook$hw1)
```

```
[1] 0.4250204
```

```r
gradebook[is.na(gradebook)]<- 0
apply(gradebook, 2, cor, y = overall_grades)
```

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

```r
which.max(apply(gradebook, 2, cor, y = overall_grades))
```

```
hw5
  5
```