Class 06: R Functions Lab

Alisa Zhang (PID: A18299618)

This week we are introducing *R Functions* and how to write our own R functions.

Questions to answer:

> 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]

```
# 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)
```

Follow the guidelines from class

- Write a working snipet of code that solves a simple problem

```
# mean()
student1 <- c(100, 100, 100, 100, 100, 100, 100, 90)

mean(student1)
```

```
[1] 98.75
```

BUT we need to drop the lowest score. First we need to identidy the lowest score.

```
# Which element of the vector is the lowest
which.min(student1)
```

```
[1] 8
```

What i want to do is to *drop* (i.e. exclude) this lowest score from my mean() calculation.

```
# This will return everything but the 8th element in the vector
student1[-8]
```

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

Now we can use the answer from which.min() to return all other elements of the vector.

```
# This is our first working snippet
mean( student1[-which.min(student1)] )
```

```
[1] 100
```

What about other example students? Will this work?

We could try using the na.rm=TRUE argument for mean but this is unfair.

```
student2 <- c(100, NA, 90, 90, 90, 90, 97, 80)
mean(student2, na.rm=TRUE)
```

```
[1] 91
```

```
student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
mean(student3, na.rm=TRUE)
```

```
[1] 90
```

Another approach is to mask (i.e. replace) all NA values with 0s.

First we need to find the NA elements of the elements.

```
student2 <- c(100, NA, 90, 90, 90, 90, 97, 80)
x <- student2

is.na(x)
```

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

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

```
[1] 2
```

Now we have identified the NA elements we want to “mask” them. How to replace them with zero?

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

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

```
mean(x)
```

```
[1] 79.625
```

Recall that we should drop the lowest score…

```
x[is.na(x)] <- 0
mean( x[-which.min(x)] )
```

```
[1] 91
```

Now we are here with our code:)

```
student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
x <- student3
x[is.na(x)] <- 0
mean( x[-which.min(x)] )
```

```
[1] 12.85714
```

Now we make our function

Take the snippet and turn it into a function

Recall that every function has three parts: - A name, in our case `grade()` - Input arguments, a vector of student scores - The body i.e. our working snippet of code

Using Studio I will select ‘Code > Extract Function’

```r
grade <- function(x) {
  x[is.na(x)] <- 0
  mean( x[-which.min(x)] )
}
```

```r
grade(student1)
```

```
[1] 100
```

```r
grade(student2)
```

```
[1] 91
```

```r
grade(student3)
```

```
[1] 12.85714
```

This looks great! We now need to add comments to explain this to our future selves and others who want to use this function.

```r
#' Calculate the average score for a vector of a student's score dropping
#' the lowest. Missing value would be replaced with zero.
#'
#' @param x A numeric vector of hw scores
#'
#' @returns Average score
#' @export
#'
#' @examples
#'  student <- c(100, NA, 90, 97)
#'  grade(student)
#'

grade <- function(x) {
  # Mask NA with zeros
  # Treat missing values as zeros
  x[is.na(x)] <- 0
  # Exclude lowest score from mean
  mean( x[-which.min(x)] )
}
```

Now finally we can use our function on real class data from this CSV format: "https://tinyurl.com/gradeinput"

```r
url <- "https://tinyurl.com/gradeinput"
gradebook <- read.csv(url, row.names = 1)
```

```r
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]

To answer question2, we can run the apply() function and save the result.

```r
result <- apply(gradebook, 1, grade)
which.max(result)
```

```
student-18
        18
```

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

```r
ave.score <- apply(gradebook, 2, mean, na.rm=TRUE)
ave.score
```

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

```r
which.min(ave.score)
```

```
hw3
  3
```

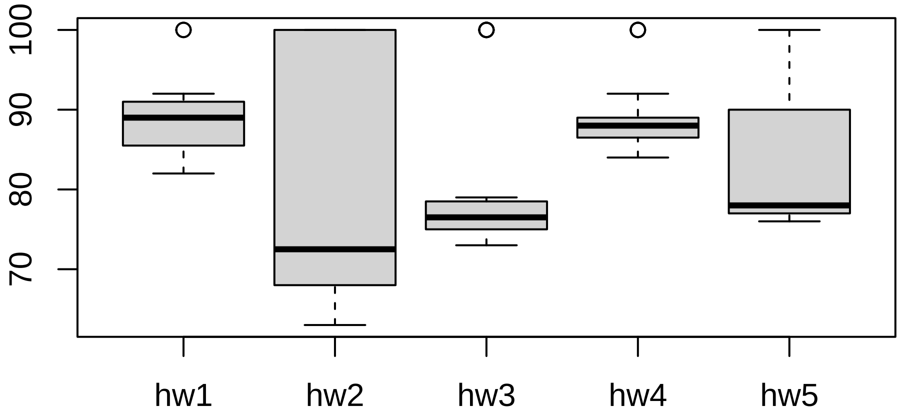
```
med.score <- apply(gradebook, 2, median, na.rm=TRUE)
med.score
```

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

```
which.min(med.score)
```

```
hw2 
  2 
```

```
boxplot(gradebook)
```



So homework 2 is probably the lowest.

> 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]

Are the final result (i.e. average score of each student) correlated with the result (i.e. score) of individual hws - the gradebook columns

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

```r
cor(result, masked.gradebook$hw5)
```

```
[1] 0.6325982
```

```r
apply(masked.gradebook, 2, cor, x=result)
```

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

Q5. Make sure you save your Quarto document and can click the “Render” (or Rmark- down”Knit”) button to generate a PDF foramt report without errors. Finally, submit your PDF to gradescope. [1pt]