R Functions Lab Class 6

Carolina Merino

All functions in R have at least 3 things:

- A **name**, we pick this and use it to call the function.
- Input **arguments**, there can be multiple comma seperated inputs to the function.
- The **body**, lines of R code that do the work of the function.

Our first wee function:

```
add <- function(x) {
  x + 1
}
```

Let's test our function

```
add <- function(x, y = 0) {
sum(x) + y
}
```

```
add(c(1, 2, 3), y = 10)
```

```
[1] 16
```

```
add(10)
```

```
[1] 10
```

Let's try something more interesting. Make a sequence generation tool.

The `sample()` function could be useful here.

```
sample(1:10, size = 3)
```

```
[1]  8 10  4
```

Change this to work with nucleotides A C G and T and return a 3 of them

```
n <- c("A", "C", "G", "T")
sample(n, size = 15, replace = TRUE)
```

```
 [1] "T" "T" "G" "C" "T" "A" "A" "A" "T" "G" "A" "T" "G" "G" "A"
```

Turn this snippet into a function that returns a user specificied length dna sequence. Let's call it `generate_dna()`...

```
# This function is to generate a random DNA sequence
generate_dna <- function(length) {
   nucleotides <- c("A", "C", "G", "T")
   dna_vector <- sample(nucleotides, size = length, replace = TRUE)

  # Optional: combine into a single string for easier viewing
  dna_sequence <- paste(dna_vector, collapse = "")

  return(dna_sequence)
}

# See data for 15 dna sequence and for 25 dna sequence
generate_dna(15)
```

```
[1] "ACCGCGAGTCCTGGG"
```

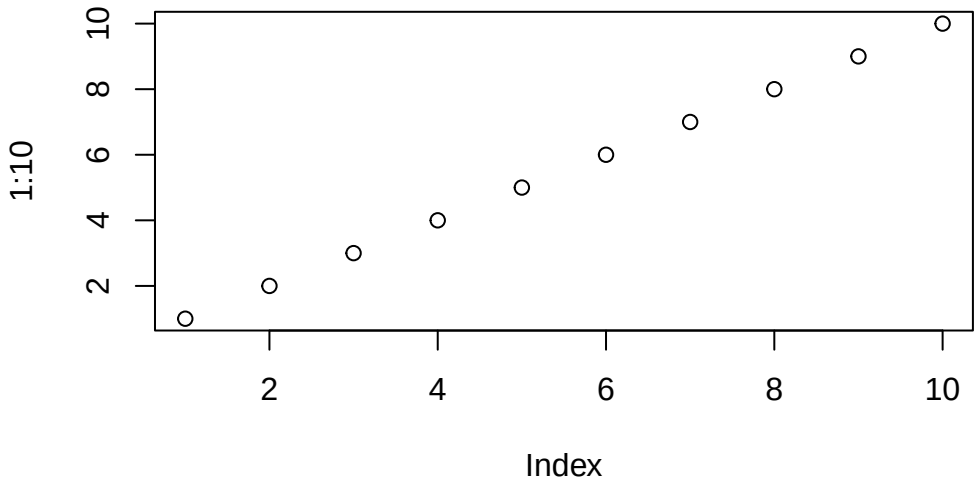
```
generate_dna(25)
```

```
[1] "ATGTGGCTAGAACCGCTATGCGACT"
```

Questions to answer:

Q1. This is a question 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]

```
plot(1:10)
```




```
# 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 sovles a simple problem

```
#Straight forward 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 identify the lowest score.

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

```
[1] 8
```

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

```
#This will return everything, but the eights
#element of 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 the other exmaple students? Will this work for them?

We could try using na.rm=TRUE argument for mean but this is pants! Not a good approach i.e. 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 zero

First we need to find the NA elements of the vector. How do we find the NA 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. Replace them with zero?

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

```
[1] NA
```

Instead we will make the NA elements zero

```
#This is useful!
x[is.na(x)] <-0
x
```

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

```
mean(x)
```

```
[1] 79.625
```

Recall we should drop the lowest score now...

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

```
[1] 91
```

Now we are essentially there with our working snippet!

```
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 into a function Every function has 3 parts

-A name, in our case `grade()` -Input arguments, a vector of student scores -The body i.e. our working snippet of code

Using RStUDION I will select `Code > Extract Function`

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

```
grade(student1)
```

```
[1] 100
```

```
grade(student2)
```

```
[1] 91
```

```
grade(student2)
```

```
[1] 91
```

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

```
#' Calculate the average score for a vector of
#' students scores dropping the lowest score.
#' Missing values will be treated as zero.
#'
#' @param x A numeric vector of homework scores
#'
#' @returns Average score
#' @export
#'
#' @examples
#' student <- c(100, NA, 90, 97)
#' grade(student)
#'
#'
grade <- function(x) {
  #mask NA with zero
  #Treat missing values as zero
  x[is.na(x)] <- 0
  #Exclude lowest score from mean
  mean(x[-which.min(x)])
}
```

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

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

```
apply(gradebook[, sapply(gradebook, is.numeric)], 1, grade)
```

```
 [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
```

> Q2. Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook? [3pts]

To answer this we run the apply() function and save the results.

```
results <- apply(gradebook[, sapply(gradebook, is.numeric)], 1, grade)
sort(results, decreasing = TRUE)
```

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

```
which.max(results)
```

```
[1] 18
```

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

```
# Compute average scores only for numeric columns
ave.scores <- apply(gradebook[, sapply(gradebook, is.numeric)], 2, mean)

# Display average scores
ave.scores
```

```
 hw1  hw2  hw3  hw4  hw5
89.0   NA 80.8   NA   NA
```

```
# Find which column has the lowest average score
which.min(ave.scores)
```

```
hw3
  3
```

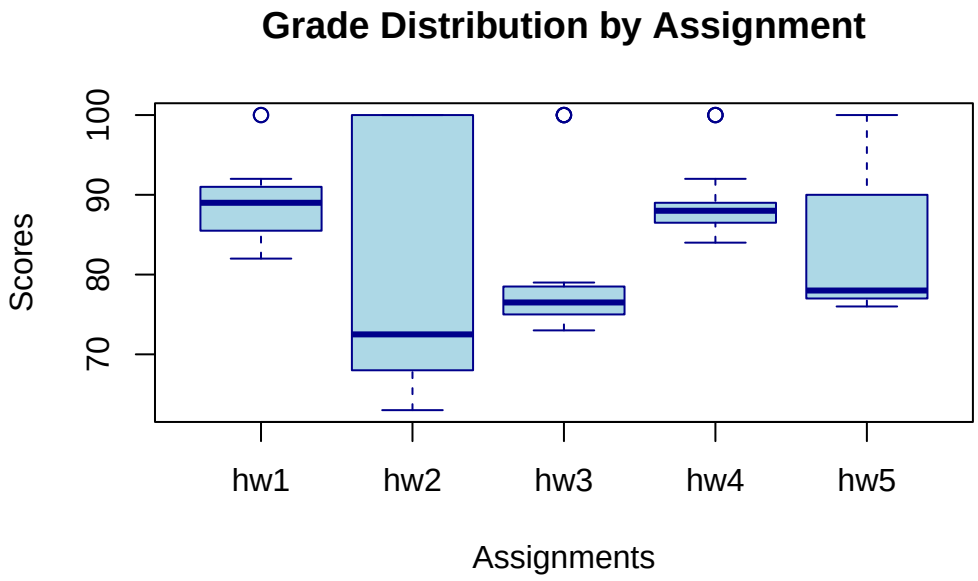
```
# Compute median scores only for numeric columns
med.scores <- apply(gradebook[, sapply(gradebook, is.numeric)], 2, median)

# Display median scores
med.scores
```

```
 hw1  hw2  hw3  hw4  hw5
89.0   NA 76.5   NA   NA
```

```
# Find which column has the lowest median score
which.min(med.scores)
```

```
hw3
  3
```

```r
# Make a boxplot using only numeric columns
boxplot(gradebook[, sapply(gradebook, is.numeric)],
        main = "Grade Distribution by Assignment",
        ylab = "Scores",
        xlab = "Assignments",
        col = "lightblue",
        border = "darkblue")
```



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]

```r
# Keep only numeric columns (homework scores)
numeric_grades <- gradebook[, sapply(gradebook, is.numeric)]
```

```r
# Calculate average score per student using our grade() function
average_score <- apply(numeric_grades, 1, grade)
```

```r
# Solve for a correlation of each homework with the average score
correlations <- sapply(numeric_grades, function(hw) cor(hw, average_score, use="complete.obs
correlations  # see all correlations
```

```
        hw1         hw2         hw3         hw4         hw5
0.42502036  0.61142768  0.30425610 -0.09644108  0.60398041
```

```r
# Find homework with highest correlation
most_predictive_hw <- names(which.max(correlations))
most_predictive_hw  # display the homework
```

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

We calculated the average grade for each student using the `grade() function`, which drops the lowest homework score. Then, we measured the correlation between each homework assignment and the students average scores. Homework hw2 had the highest correlation, meaning it was the most predictive of overall performance.

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