# Introduction to R: Assignment 1

Guilherme W. F. Barros

08/09/2023

## Instructions

A `.R` file with your solutions should be uploaded to Canvas by latest September 22th at 6pm. In addition, you can check your `.R` files containing the code needed to produce your solution on the website: https://statistics-umu.shinyapps.io/introduction-to-r/.

We highly recommend doing this test in order to guarantee that your solutions are correct! Note that testing a file is not the same as a Canvas submission.

The aim of this course is for you to enhance your R programming skills, and learning by doing is really the way to get there. That said, you are encouraged to discuss course material with each other. However, the solutions you hand in should be your own work.

At last, most of the problems can have several correct ways to solve them. For example, several of the problems in this assignment can be solved with both while and for loops or even with a smart indexing of data frames, without the use of any loops.

**When you are done, submit your .R file in Canvas with the name Assignment_1_id.R (for example, Assignment_1_guwa0049.R)**

## Code examples

These are some example code that you may modify to your own purposes in order to solve the assignment problems.

### Example 1: If/else combination with a for loop

This code samples 50 random numbers between 1 and 1000 and then separates them into odd and even numbers.

```
vector_numbers = sample(1:1000, size = 50)
odd_numbers <- c()
even_numbers <- c()

for(index in 1:length(vector_numbers)){
  # The modulus operator %% gives the remainder of a division. So 3 %% 2 = 1 but 4 %% 2 = 0
  if(vector_numbers[index] %% 2 == 0){
    even_numbers <- c(even_numbers,vector_numbers[index])
  }else{
    odd_numbers <- c(odd_numbers,vector_numbers[index])
  }
}
}
```

**Example 2: Manipulating a data frame**

R has some built in datasets, which are frequently used for testing and studying purposes. Among these, the dataset USArrests contains statistics about violent crime rates by us state. We can add it to our environment, look at it and see the average muder rate by doing:

```
USArrests <- get(data("USArrests"))
print(USArrests)
average_murder <- mean(USArrests$Murder)
print(average_murder)
```

The code below prints the names of all the states in the dataset one by one, then adds it to a vector:

```
total_rows <- nrow(USArrests)
state_names <- row.names(USArrests)
all_states = c()

for (row in 1:total_rows){
    print(state_names[row])
  all_states = c(all_states, state_names[row])
}
```

Here are some examples of how you can also create new datasets that are just a subset of the initial dataset:

```
# For example, let's subset with only US states that border Mexico:
mexican_border <- USArrests[row.names(USArrests) %in% c('California', 'Arizona', 'New Mexico', 'Texas')
print(mexican_border)

# We can also select all states with a specific name. For example:
Utah <- USArrests[row.names(USArrests) == 'Utah',] #This is just one line, but it is an example

# Or with a specific UrbanPop. For example:
UrbanPop80 <- USArrests[ USArrests$UrbanPop == 80,]
```

## Problems to solve

In every problem, we already create an empty object that needs to be filled. Note that your answer should have an object with exactly the same name.

### Q1)

Using the USArrests dataset from the above example, write your own code (and/or modify one of the examples above) so that we create two vectors:

- The first vector should be named below_average and should contain the names of all states with a murder rate below the average (average_murder)
- The second vector should be named above_average and should contain the names of all states with a murder rate above the average (average_murder)

```
below_average <- c()
above_average <- c()
```

For Q2 and the rest of the assignment, we will be using the penguins dataset, from the package 'palmerpenguins'. We can install the package and load it by:

**Remember to comment out the install.packages line after running it once**

```
install.packages('palmerpenguins')
library(palmerpenguins)
penguins <- get(data("penguins"))

# 11 rows have incomplete information, we are interested only in the complete cases of the penguins
# Therefore, we do:

penguins <- penguins[complete.cases(penguins),]
```

### Q2)

Using the above complete penguins dataset, separate it into two datasets, one for male penguins and one for female. Use the name 'male_penguins' and 'female_penguins' for the respective dataframes

```
male_penguins <- data.frame()
female_penguins <- data.frame()
```

### Q3)

Do male or female penguins in the dataset have larger flippers for their body? Calculate the average flipper size divided by body mass (flipper_length_mm / body_mass_g) for both male and female penguins. Remember you can use the mean() function as in the example above.

Fill the resulting value in the objects flipper_proportion_male and flipper_proportion_female as a number.

Hint: You want flipper_length (in mm) / body_mass (g) for each penguin then take the mean.

```
flipper_proportion_male <- 0
flipper_proportion_female <- 0
```

### Q4)

On average, penguins have a bill (or beak) length that is more than twice of its depth. Due to genetic and species variation, this is not true for all penguins. Some penguins just have "big mouths". Or maybe very short beaks.

Create a dataframe that includes only penguins that have a bill length that is less than twice its depth. Do not include the cases that the bill length is equal to twice the depth.

Hint: You want the penguins with bill_length (in mm) $<$ 2 * bill_depth (in mm)

```
big_mouth <- data.frame()
```

**Q5)**

You might have noticed that all penguins with a "big mouth" (or a short beak) are all from the same species, "Adelie". What is the proportion of adelie penguins with a big mouth? Assign that number to the variable adelie_big_mouth_proportion.

Hint: You need the total of adelie penguins with a big mouth / total amount of adelie penguins. Do not multiply by 100 (we want the proportion, not the percentage).

```
adelie_big_mouth_proportion <- 0
```

**Q6)**

When measuring the penguins, there was a problem with the calibration of the scale in the Biscoe island. The scale used in Biscoe inflated all weights of penguins by 250g. Create a corrected dataset "corrected_penguins" with the adjusted weights of the Biscoe's penguins.

Example: if a penguin in Biscoe was weighted as 3750, you should correct it to 3750 - 250 = 3500. Note: Do not alter the original penguins dataset! You should alter only the corrected_penguins dataset

```
corrected_penguins <- penguins
```