# Assignment 2: Coding Basics

Victoria Thompson

## OVERVIEW

This exercise accompanies the lessons/labs in Environmental Data Analytics on coding basics.

## Directions

1. Rename this file `<FirstLast>_A02_CodingBasics.Rmd` (replacing `<FirstLast>` with your first and last name).
2. Change "Student Name" on line 3 (above) with your name.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file.
6. After Knitting, submit the completed exercise (PDF file) to Canvas.

## Basics, Part 1

1. Generate a sequence of numbers from one to 55, increasing by fives. Assign this sequence a name.

2. Compute the mean and median of this sequence.

3. Ask R to determine whether the mean is greater than the median.

4. Insert comments in your code to describe what you are doing.

```
#1. Using the seq function, I am inputting the "from" (1), "to" (55),
#and "by" (5) elements. Then,
#I assign those values to a variable ("sequen_to_55") using "<-".

sequen_to_55 <- seq(1, 55, 5)

#2. Using "mean" function to calculate the mean,
#and "median" to calculate the median. Saving each to
#sequen_mean and sequen_median, respectively.
sequen_mean <- mean(sequen_to_55)
sequen_median <- median(sequen_to_55)

#3. I compare the two values in an "if" statement.
#If the value of sequen_mean is greater than sequen_median, then the function
#will print the word "greater" in the console. Using "else", is
#this is not true (i.e. if it is equal to or less than sequen_median),
#then "less or equal" will be printed in the console.

if (sequen_mean > sequen_median) {
```

```
  print("greater")
} else {
  print("less or equal")
}
```

```
## [1] "less or equal"
```

## Basics, Part 2

5. Create three vectors, each with four components, consisting of (a) student names, (b) test scores, and (c) whether they are on scholarship or not (TRUE or FALSE).
6. Label each vector with a comment on what type of vector it is.
7. Combine each of the vectors into a data frame. Assign the data frame an informative name.
8. Label the columns of your data frame with informative titles.

```
#5 #6
student_names <- c("Victoria", "Audrey", "Patty", "Sam") #Character vector (char)
test_scores <- c(81,99,67,85) #Numerical vector (num)
scholarship_status <- c(TRUE,FALSE,FALSE,TRUE) #Logical vector (logi)

#7
student_scores_and_scholarships <- data.frame(student_names,test_scores,scholarship_status)

#8
names(student_scores_and_scholarships) <- c("Name","TestScores","ScholarshipStatus")
```

9. QUESTION: How is this data frame different from a matrix?

   Answer: Matrices can only contain one type of data class (ex. all numbers). This data frame is able to contain data of multiple class types that vary based on their row (ex. char, num, and logi).

10. Create a function with one input. In this function, use `if...else` to evaluate the value of the input: if it is greater than 50, print the word "Pass"; otherwise print the word "Fail".
11. Create a second function that does the exact same thing as the previous one but uses `ifelse()` instead if `if...else`.
12. Run both functions using the value 52.5 as the input
13. Run both functions using the **vector** of student test scores you created as the input. (Only one will work properly...)

```
#10. Create a function using if...else

test_over_50_v1 <- function(x) {
  if (x > 50) {
    return("Pass")
  } else {
    return("Fail")
```

```
  }
}
#11. Create a function using ifelse()

test_over_50_v2 <- function(x){
  ifelse(x>50, return("Pass"),return("Fail"))
}

#12a. Run the first function with the value 52.5

test_over_50_v1(52.5)
```

```
## [1] "Pass"
```

```
#12b. Run the second function with the value 52.5

test_over_50_v2(52.5)
```

```
## [1] "Pass"
```

```
#13a. Run the first function with the vector of test scores

  #test_over_50_v1(test_scores)

#13b. Run the second function with the vector of test scores

test_over_50_v2(test_scores)
```

```
## [1] "Pass"
```

14. QUESTION: Which option of `if...else` vs. `ifelse` worked? Why? (Hint: search the web for "R vectorization") > Answer: The function that used 'ifelse' worked, while the function with 'if...else' did not. This is because of R vectorization, which means that operations occur in parallel in certain R objects. 'if...else' is not vectorized, as it is designed for straightforward, single-input operations. When a vector is put into 'if... else', the function returns an error as it cannot parallel compute multple input values. 'ifelse' is able to evaluate each value in a vector, which is why it worked with the input vectors.

**NOTE** Before knitting, you'll need to comment out the call to the function in Q13 that does not work. (A document can't knit if the code it contains causes an error!)