Assignment 2: Coding Basics

Brianna Karson

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.
7. Initial here to acknowledge that you did not use AI at all in completing this assignment: BK

Basics, Part 1

1. Use R's `seq()` function to create a sequence of numbers from 100 to 333, increasing by threes. Assign this sequence a variable name.
2. Compute the *mean* of this sequence, assigning this values its own variable name.
3. Compute the *standard deviation* (`sd()`) of this sequence, assigning this values its own variable name.
4. Display the the mean minus the standard deviation as well as the mean plus the standard deviation.
5. Insert comments in your code to describe what you are doing.

```
#1. #sequence function
three_sequence <- seq(100,333,3)

#2. #generating mean
mean_three_sequence <- mean(three_sequence)
mean(three_sequence)
```

```
## [1] 215.5
```

```r
#3. #computing standard deviation
st_dev_three_sequence <- sd(three_sequence)
sd(three_sequence)
```

```
## [1] 67.98162
```

```r
#4.
#calculating mean minus st dev
mean_three_sequence - st_dev_three_sequence
```

```
## [1] 147.5184
```

```r
#calculating mean plus st dev
mean_three_sequence + st_dev_three_sequence
```

```
## [1] 283.4816
```

Basics, Part 2

6. 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).
7. Label each vector with a comment on what type of vector it is.
8. Combine each of the vectors into a data frame. Assign the data frame an informative name.
9. Label the columns of your data frame with informative titles.

```r
#6.
#making vectors
vectorA_names <- c("Matthew","Mark","Luke","John") #character vector

vectorB_scores <- c(100,96,98,94) #numeric vector

vectorC_scholarship <- c(TRUE, FALSE, TRUE, FALSE) #logical vector

#making a data frame
student_data <- data.frame("name"=vectorA_names,"score"=vectorB_scores,
                           "scholarship"=vectorC_scholarship)
```

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

    Answer: Matrices and data frames are both two-dimensional, but a matrix can only contain one type of data. As seen above, a data frame may contain many different types of data, and each column is a named vector.

Basics, Part 3

11. 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, it returns the word "Pass"; otherwise it returns the word "Fail".
12. Create a second function that does the exact same thing as the previous one but uses `ifelse()` instead if `if...else`.
13. Run both functions using the value 54 as the input
14. Run both functions using the **vector** of student test scores you created as the input. (Only one will work properly...)

```
#11. Create a function using if...else
function_11 <- function(x) { if (x>50) {"PASS"} else {"FAIL"} }

#12. Create a function using ifelse()
function_12 <- function(x){ifelse(x>50,"PASS","FAIL")}

#13a. Run the first function with the value 54
function_11(54) #"PASS"
```

```
## [1] "PASS"
```

```
#13b. Run the second function with the value 54
function_12(54) #"PASS"
```

```
## [1] "PASS"
```

```
#14a. Run the first function with the vector of test scores
#function_11(vectorB_scores) #did not work

#14b. Run the second function with the vector of test scores
function_12(vectorB_scores) # worked
```

```
## [1] "PASS" "PASS" "PASS" "PASS"
```

15. QUESTION: Which option of `if...else` vs. `ifelse` worked? Why? (Hint: search the web for "R vectorization"; it's ok here if an AI response is presented in the search response.)

    Answer: 'if...else' did not work because this is not a vectorized function, meaning that it can only be applied to one value at a time. 'ifelse' worked because it is a vectorized function and therefore can be applied to evaluate the entire scores vector at once.

NOTE Before knitting, you'll need to comment out the call to the function in Q14 that does not work. (A document can't knit if the code it contains causes an error!)