Assignment 2: Coding Basics

Yangsen Zhang

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: Yangsen Zhang

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.

```r
#1. Create a sequence of numbers from 100 to 333, increasing by threes
seq_A2 <- seq(from = 100, to = 333, by=3)
#2. Compute the mean of this sequence
sep_A2_mean <- mean(seq_A2)
#3. Compute the standard deviation of this sequence
seq_A2_sd <- sd(seq_A2)
#4.Display the the mean minus the standard deviation as well as the mean plus the standard deviation.
sep_A2_mean - seq_A2_sd
```

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

```
sep_A2_mean + seq_A2_sd
```

```
## [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.

```
#6,7 Create three vectors, each with four components, consisting of (a) student names, (b) test scores,
student_names <- c("Mary","Bob","Tom","Judy") #character vector
test_scores <- c(99,88,77,66) #numeric vector
scholarship <- c(TRUE,TRUE,FALSE,FALSE) #logical vector

#8 Combine each of the vectors into a data frame
student_data <- data.frame(student_names,test_scores,scholarship)

#9 Label the columns of your data frame with informative titles.
colnames(student_data) <- c("student_names","test_scores","scholarship")

student_data
```

```
##   student_names test_scores scholarship
## 1          Mary          99        TRUE
## 2           Bob          88        TRUE
## 3           Tom          77       FALSE
## 4          Judy          66       FALSE
```

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

    Answer:Data frame can contain different data types in different columns but a matrix can only contain one data type.

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 to check if the value is greater than 50
Pass_Fail_if_else <- function(x) {if(x > 50) {return("Pass")} else {return("Fall")} }
#12. Create a function using ifelse()
Pass_Fail_ifelse <- function(x) {ifelse(x > 50, "Pass", "Fall")}
#13a. Run the first function with the value 54
Pass_Fail_if_else(54)
```

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

```
#13b. Run the second function with the value 54
Pass_Fail_ifelse(54)
```

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

```
#14a. Run the first function with the vector of test scores
#Pass_Fail_if_else(test_scores) commented out because it does not work
#14b. Run the second function with the vector of test scores
Pass_Fail_ifelse(test_scores)
```

```
## [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:`ifelse` worked because it is vectorized, meaning that operations occur in parallel.

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!)