Assignment 2: Coding Basics

Haochuan Zhan

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: HZ_____

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 from 100 to 333 increasing by 3
num_seq <- seq(100, 333, by = 3)
#2. Compute the mean of the sequence
seq_mean <- mean(num_seq)
seq_mean
```

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

```r
#3. Compute the standard deviation of the sequence
seq_sd <- sd(num_seq)
seq_sd
```

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

```
#4. Display mean - sd and mean + sd
seq_mean - seq_sd
```

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

```
seq_mean + seq_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.
student_names <- c("Jayson", "Kyrie", "Cooper", "Zion") #character vector
student_names
```

```
## [1] "Jayson" "Kyrie"  "Cooper" "Zion"
```

```
test_scores <- c(88, 90, 92, 76)                       #numeric vector
test_scores
```

```
## [1] 88 90 92 76
```

```
scholarship <- c(TRUE, TRUE, TRUE, FALSE)         #logical vector
scholarship
```

```
## [1]  TRUE  TRUE  TRUE FALSE
```

```
#8.
students_info <- data.frame(student_names, test_scores, scholarship, stringsAsFactors = FALSE)
#9.
names(students_info) <- c("Name", "Score", "Scholarship")
students_info
```

```
##     Name Score Scholarship
## 1 Jayson    88        TRUE
## 2  Kyrie    90        TRUE
## 3 Cooper    92        TRUE
## 4   Zion    76       FALSE
```

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

    Answer:A data frame can contain different data types in different columns (e.g., character, numeric, logical), while a matrix must contain only 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
pass_fail_1 <- function(x) {if (x > 50) "Pass" else "Fail"}
#12. Create a function using ifelse()
pass_fail_2  <- function(x) {ifelse(x > 50, "Pass", "Fail")}
#13a. Run the first function with the value 54
pass_fail_1(54)
```

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

```
#13b. Run the second function with the value 54
pass_fail_2(54)
```

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

```
#14a. Run the first function with the vector of test scores
#pass_fail_1(test_scores)
#14b. Run the second function with the vector of test scores
pass_fail_2(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 with the vector because it is vectorized and evaluates each element of the vector, returning element-wise results. While if...else expects a single TRUE/FALSE condition (length 1)

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!)