Assignment 2: Coding Basics

Yufan Du

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.  Generate a sequence of numbers from one to 55, increasing by fives. Assign this sequence a name.
number_sequence <- seq(1, 55, by = 5)

#2.  Compute the mean and median of the sequence
mean_value <- mean(number_sequence)
median_value <- median(number_sequence)

#3. Ask R to determine whether the mean is greater than the median
mean_is_greater <- mean_value > median_value

# 4. Print out the results to check
print(paste("Mean:", mean_value))
```

```
## [1] "Mean: 26"
```

```
print(paste("Median:", median_value))
```

```
## [1] "Median: 26"
```

```r
print(paste("Is the mean greater than the median?", mean_is_greater))
```

```
## [1] "Is the mean greater than the median? FALSE"
```

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.

```r
# 5. Create three vectors with four components
# (a) Vector of student names (character vector)
student_names <- c("Marina", "Alex", "Yufan", "Thomas")

# (b) Vector of test scores (numeric vector)
test_scores <- c(99, 99, 77, 44)

# (c) Vector of scholarship status (logical vector)
scholarship_status <- c(TRUE, TRUE, FALSE, FALSE)

# 7. Combine the vectors into a data frame and assign it an informative name
student_data <- data.frame(Name = student_names, Score = test_scores, Scholarship = scholarship_status)

# 8. Label the columns of your data frame with informative titles
colnames(student_data) <- c("Student Name", "Test Score", "On Scholarship")

# Print the data frame to check
print(student_data)
```

```
##   Student Name Test Score On Scholarship
## 1       Marina         99           TRUE
## 2         Alex         99           TRUE
## 3        Yufan         77          FALSE
## 4       Thomas         44          FALSE
```

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

   Answer: A matrix is a two-dimensional array where all elements must be of the same type. But this data frame has different types of data in each column. For example, in this data frame, one column contains character strings (student names), another contains numeric values (test scores), and another contains logical values (scholarship status).

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
check_pass_ifelse <- function(score) {
  if (score > 50) {
    print("Pass")
  } else {
    print("Fail")
  }
}

#11. Create a function using ifelse()
check_pass_ifelse_vector <- function(score) {
  result <- ifelse(score > 50, "Pass", "Fail")
  print(result)
}

#12a. Run the first function with the value 52.5
check_pass_ifelse(52.5)
```

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

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

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

```
#13a. Run the first function with the vector of test scores
# check_pass_ifelse(test_scores) #This one doesn't work

#13b. Run the second function with the vector of test scores
check_pass_ifelse_vector(test_scores)
```

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

14. QUESTION: Which option of `if...else` vs. `ifelse` worked? Why? (Hint: search the web for "R vectorization")

    Answer:The ifelse option worked when applied to the vector of student test scores. This is because ifelse() is vectorized, meaning it can operate on each element of a vector and return a vector of results. In contrast, the if…else structure is designed for single condition checks and doesn't naturally handle vectors. Therefore, when if…else was used with a vector, it didn't work correctly, leading to errors or unintended behavior. Vectorization allows ifelse() to efficiently apply the condition across all elements of the vector simultaneously, making it the preferred choice for operations on vectors in R.

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!)