Assignment 2: Coding Basics

Jingze Dai

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. The sequence will be named "seq_55", made up of a list of numbers as numeric values from 1 to 55

seq_55 <- c(1:55)

#2. We can use mean and median command to obtain the desired values

# calculating mean
mean_seq <- (mean(seq_55))
# calculating median
median_seq <- median(seq_55)

# printing results
paste("The mean of the sequence is", mean_seq)
```

```
## [1] "The mean of the sequence is 28"
```

```
paste("The median of the sequence is ", median_seq)
```

```
## [1] "The median of the sequence is  28"
```

```
#3. We can subtract the median with the mean and see if the result is larger than zero

# calculating the difference
mean_median_comparison <- mean_seq - median_seq

# if function to compare the difference
if (mean_median_comparison >0) {
  print("The mean is greater than the median")
} else {
  print("The mean is not greater than the median")
}
```

```
## [1] "The mean is not greater than the median"
```

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. Randomly generating three students

student_name <- c("Jingze", "Dom", "Ryosuke", "Max") # type of vector is characters
test_score <- c(25, 21, 27, 30) # type of vector is numeric
scholarship_status <- c(FALSE, TRUE, FALSE, TRUE) # type of vector is logical

#7. Combining vectors; the data frame is named as student_info_df

student_info_df <- data.frame(student_name, test_score, scholarship_status)

#8. Labeling the columns of the data frame

names(student_info_df) <- c("Student Name","Test Score out of 30","Scholarship Status")

# checking if the title changed
summary(student_info_df)
```

```
##  Student Name       Test Score out of 30 Scholarship Status
##  Length:4           Min.   :21.00        Mode :logical
##  Class :character   1st Qu.:24.00        FALSE:2
##  Mode  :character   Median :26.00        TRUE :2
##                     Mean   :25.75
##                     3rd Qu.:27.75
##                     Max.   :30.00
```

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

Answer: Data frame can store data of different types, for example, student name is a factor, test score is numerical and scholarship status is logical. However, a matrix can only store data of the same type, thus a matrix is homogeneous and a data frame is heterogeneous.

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...)

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

above_50_if <- function(x) {
  if (x > 50) {
    print("Pass")
  } else {
    print("Fail")
  }
}

#11. Create a function using ifelse()

above_50_ifelse <- function(x) {
  result <- ifelse(x > 50, "Pass", "Fail")
  print(result)
}

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

above_50_if(52.5)
```

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

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

above_50_ifelse(52.5)
```

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

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

# this will return an error of "Error in if (x > 50) { : the condition has length > 1"
# to successfully knit the pdf, the line of code is commented out

#above_50_if(test_score)

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

above_50_ifelse(test_score)
```

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

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

    Answer: `ifelse` worked, because it is designed to be vectorized, meaning that it can take vectors as test arguments and will return a vector of the same length as the input. However, `if...else` is not vectorized, because it is designed to process single logical conditions instead of a vector of conditions.

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!)