Assignment 2: Coding Basics

Leah Li

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 1 to 55, increasing by 5
my_sequence <- seq(1, 55, 5)
my_sequence
```

```
##  [1]  1  6 11 16 21 26 31 36 41 46 51
```

```
#2.Calculate the mean of the sequence
mean_seq <- mean(my_sequence)
#Calculate the median of the sequence
median_seq <- median(my_sequence)
#Print results
mean_seq
```

```
## [1] 26
```

```
median_seq
```

```
## [1] 26
```

```r
#3.#Check if the mean is greater than the median of this sequence, assign result to mean_median_compare
mean_median_compare <- mean_seq > median_seq
#Print results
mean_median_compare
```

```
## [1] 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. A Vector of student names;
#6. Vector type: Character
student_names <- c("John", "Elsa", "Snow", "Minnie")
class(student_names)
```

```
## [1] "character"
```

```r
#5.A Vector of test scores;
#6.Vector type: Numeric
test_scores <- c(60, 90, 87, 100)
class(test_scores)
```

```
## [1] "numeric"
```

```r
#5. A Vector that indicates whether the students are on scholarship or not;
#6. Vector type: Logical
scholarship_recipient <- c(TRUE, FALSE, TRUE, FALSE)
class(scholarship_recipient)
```

```
## [1] "logical"
```

```r
#7. Combine the vectors into a data frame, assign the data frame an informative name.
student_data <- data.frame(student_names,
                           test_scores,
                           scholarship_recipient)

#8. Label the columns of your data frame with informative titles
student_data <- data.frame(
  "Student Name" = student_names,
  "Test Score" = test_scores,
  "Scholarship" = scholarship_recipient
)

# Display the result
student_data
```

```
##   Student.Name Test.Score Scholarship
## 1         John         60        TRUE
## 2         Elsa         90       FALSE
## 3         Snow         87        TRUE
## 4       Minnie        100       FALSE
```

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

   Answer: A data frame can have different data types in each column. For instance, the student_data data frame has character data (student names), numeric data (test scores), and logical data (scholarship status). However, a matrix can only have one data type for all its elements.

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 to evaluate student scores

evaluate_score <- function(student_score) {
  if (student_score > 50) {
    print("Pass")
  }
  else {
    print("Fail")
  }
}

#11. Create a function using ifelse()

evaluate_score2 <- function(student_score) {
  ifelse(student_score > 50, "Pass", "Fail")
}

#12a. Run the first function with the value 52.5
Run_first_function <- evaluate_score(52.5)
```

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

```
Run_first_function
```

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

```
#12b. Run the second function with the value 52.5
Run_second_function <- evaluate_score2(52.5)
Run_second_function
```

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

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

#This is the code I am trying to use. But it seems to be not working: Run_first_function_vector <- eval

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

Run_second_function_vector <- evaluate_score2(test_scores)
Run_second_function_vector
```

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

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

    Answer:The option of 'ifelse' worked. Upon searching the R vectorization, I found out that the function 'ifelse' is vectorized, meaning that this function can apply a single operation to each element of a vector. This is very useful since it could help us handle vectors which could have multiple values. In contrast, the if...else structure is not vectorized and it can only evaluate the first value of the vector. That is why it does not work in this case because we are giving it a vector to evaluate.

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!)