Assignment 2: Coding Basics

Sophie Valkenberg

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.
seq_55_by5 <- seq(1,55,5) #I defined the object "seq_55_by5" by creating the name, then using the seque
seq_55_by5 #here, I recalled the object. to check the sequence is correct
```

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

```
#2.
mean <- mean(seq_55_by5) #mean is 26. I used the function "mean" to find the mean and created an object
median <- median(seq_55_by5) #median is 26. I used the function "median" to find the median and created

#3.
mean > median #here, I used a logical expression to determine if the mean is greater than the median. T
```

```
## [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.

```
vector_StudentNames <- c("Brynn", "Ky", "Fiona", "Andreana") #Vector type: Character
vector_TestScores <- c(95, 82, 78, 87) #Vector type: Numeric
vector_Scholarship <- c(TRUE, TRUE, FALSE, FALSE) #Vector type: Logical

dataframe_StudentInfo <- data.frame(vector_StudentNames, vector_TestScores, vector_Scholarship) #here,
names(dataframe_StudentInfo) <- c("Name", "Test Score", "Scholarship") #Here, I labeled the columns wit
```

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

   Answer: This data frame is different from a matrix because it contains elements of different types. The three vectors that were combined to create this data frame had charcter, numeral, and logical data. In a matrix, all of the elements would need to be of the same type.

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

```
## function(x) {
##   if(x > 50) {print("Pass")}
##   else({print("Fail")})
##   }
```

```
#11. Create a function using ifelse()
(evaluate_passing_ifelse <- function(x) {
  ifelse(x > 50, "Pass", "Fail")
})
```

```
## function(x) {
##   ifelse(x > 50, "Pass", "Fail")
## }
```

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

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

```r
#Printed back word "Pass"

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

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

```r
#Printed back word "Pass"

#13a. Run the first function with the vector of test scores
#(evaluate_passingVector <- function(vector_TestScores) {
 # if(x > 50) {print("Pass")}
 # else({print("Fail")})
 # })
#evaluate_passingVector (vector_TestScores)

#13b. Run the second function with the vector of test scores
(evaluate_passing_ifelseVector <- function(x) {
  ifelse(x > 50, "Pass", "Fail")
})
```

```
## function(x) {
##   ifelse(x > 50, "Pass", "Fail")
## }
```

```r
evaluate_passing_ifelseVector(vector_TestScores)
```

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

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

    Answer: Between the "if...else" and "ifelse" functions, the "ifelse" function was the only on that worked for the Test Scores vector. From reading Yale's "R for Novices" page R-Bloggers, I learned that most functions in R are vectorized. This means that the function will act on each element of the vector at once without needing to repeat the function on each element individually. I also learned that "ifelse" is a vectorized version of "if...else," so it makes sense that the non-vectorized version would not work on a vector.

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!)