Assignment 2: Coding Basics

Ellie Harrigan

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.

```
seq(1,55,5)
```

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

```
fifty_five_sequence <- seq(1,55,5)
fifty_five_sequence
```

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

```
#1. from, to, by, then assigned sequence a name, tested name to make sure it was assigned properly
mean(fifty_five_sequence)
```

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

```
median(fifty_five_sequence)
```

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

```r
#2. computed summary statistics of the sequence
mean(fifty_five_sequence) > median(fifty_five_sequence)
```

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

```r
mean(fifty_five_sequence) == median(fifty_five_sequence)
```

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

```r
#3. Asked conditional statements to compare the mean and median. mean_seq > median_seq "false"
#then asked mean_seq == median_sequence output is "true"
```

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
#create three vectors, each with four components
a <- c("Jo", "Liz", "Sara", "Jack") # character vector containing student names,
b <- c(100,90,80,70) # numeric vector test scores for each student
c <- c(TRUE, FALSE, TRUE, FALSE) #logical vector indicating scholarship for each student

#combine vectors into a data frame and assign the data frame an informative name
student_data <- data.frame(a,b,c) #combined data frame the following vectors with name student_data

#set the data frame column names
names(student_data) <- c("Student_Names", "Test_Scores", "Scholarship_Status")
#replaced a,b,c with informative titles
print(student_data)
```

```
##   Student_Names Test_Scores Scholarship_Status
## 1            Jo         100               TRUE
## 2           Liz          90              FALSE
## 3          Sara          80               TRUE
## 4          Jack          70              FALSE
```

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

   Answer: Matrices have columns that can contain only one data type while a data frame can have different data types, such as numeric values, charater values, and logical values.

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
# followed steps shown in the Rstudio Basics, part 3 video.
#Around 10 minutes in when video goes over functions as recipes.
#The pass_fail is a function with outputs based on conditional statements, True="Pass"/ False="Fail"
pass_fail <- function(x) {
  if (x > 50) {
    print("Pass")
  } else("Fail")
}
#11. Create a function using ifelse()
 # prompted AI to help me understand how to write the code for the function ifelse() - result is below.
#The ifelse function is vectorized.
check_pass_fail_ifelse <- function(x) {
  result <- ifelse(x > 50, "Pass", "Fail") #ifelse evalutes condition x > 50, if x > 50 True="Pass";
  #if x > 50 False="Fail"
  print(result)
}
#12a. Run the first function with the value 52.5
pass_fail(52.5)
```

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

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

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

```
#13a. Run the first function with the vector of test scores
test_scores <- c(100,90,80,70)
test_scores
```

```
## [1] 100  90  80  70
```

```
# commenting out this line pass_fail(test_scores)

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

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

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

    Answer: ifelse worked because this is already an existing function in R that is vectorized.

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!)