Assignment 2: Coding Basics

Aishwarya Patankar

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. Generating a sequence x Using the seq Function to generate a sequence of numbers with a gap of 5.
x <- seq(1,55,5)
x
```

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

```
#2. Using mean function and median function to find the average of the numbers in the sequence x.
average_x <- mean(x)
median_x <- median(x)

#3. Using == to find the logical output to determine if mean and median of sequence x is the same.
average_x ==median_x
```

```
## [1] 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
#5.
#creating vector with four student names #Type: Character
student_names <- c("moana","belle","merida","elsa")
student_names
```

```
## [1] "moana"  "belle"  "merida" "elsa"
```

```r
#creating vector with four test scores # Type:Numeric
test_scores <- c(95,90,93,92)
test_scores
```

```
## [1] 95 90 93 92
```

```r
#creating vector of scholarships #Type:Logical
scholarship <- c(FALSE,TRUE, FALSE, TRUE)
scholarship
```

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

```r
#6 Checking the class types
class(student_names)
```

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

```r
class(test_scores)
```

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

```r
class(scholarship)
```

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

```r
#7
#Creating a dataframe called student_status for student names, test scores and scholarship
student_status <- data.frame(student_names,test_scores,scholarship)

#8
#Naming the dataframe coloumns with a title
#names(student_status) <- c("Name","Score", "Scholarship"); View(student_status) # This command has bee
```

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

   Answer: A dataframe can have data of multiple types shown in a tabular format whereas matrices store values 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
if(median_x > 50){
  print("Pass")
} else {
  print("Fail")
}
```

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

```
#11. Create a function using ifelse()
ifelseresult11 <- ifelse(median_x > 50, "Pass", "Fail")
ifelseresult11
```

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

```
#12a. Run the first function with the value 52.5
if(52.5 > 50){
  print("Pass")
} else {
  print("Fail")
}
```

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

```
#12b. Run the second function with the value 52.5
ifelseresult12b <- ifelse(52.5 > 50, "Pass", "Fail")
ifelseresult12b
```

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

```
#13a. Run the first function with the vector of test scores
#if(test_scores > 50){
 # print("Pass")
#} else {
 # print("Fail")
#}
```

```
# This function has been commented out since it does not work

#13b. Run the second function with the vector of test scores
ifelseresult13b <- ifelse(test_scores > 50, "Pass", "Fail")
ifelseresult13b
```

```
## [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 ifelse option works in this case because it is the vectorized alternative to the because it is the vectorized alternative to the standard if…else statement used in R. 'ifelse' allows us to apply the ifelse condition to the entire vector in one go instead of needing to loop through individual elements of the vector as in case of 'if…else'.

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!)