Assignment 2: Coding Basics

Chunyi Xu

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. Using the sequence code command to generate a sequence, while using "<-" to assign this sequence a
Fiftyfive_sequence <- seq(1, 55, 5)
Fiftyfive_sequence
```

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

```
#2. Using the mean and the median summary statistic command to compute the mean and median of the seque
mean_Fiftyfive <- mean (Fiftyfive_sequence)
median_Fiftyfive <- median (Fiftyfive_sequence)

#3. Using a conditional statement to ask R to determine if the mean "mean_Fiftyfive" is greater than th
mean_Fiftyfive > median_Fiftyfive
```

```
## [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.

```
#5. Creating three vectors with four components for each and naming them according to the information t
student_names <- c("Peter","David","Maria","Alice")
student_names
```

```
## [1] "Peter" "David" "Maria" "Alice"
```

```
test_scores <- c(94, 96, 95, 98)
test_scores
```

```
## [1] 94 96 95 98
```

```
scholarship <- c("FALSE", "TRUE", "FALSE", "TRUE")
scholarship
```

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

```
#6.Using the class command to ask R to specify what kind of vectors are created and using a comment to
class(student_names) #Vector type: character
```

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

```
class(test_scores) #Vector type: numeric
```

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

```
class (scholarship) #Vector type: character
```

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

```
#7. Combing the three vectors into one dataframe and naming it student_test_scholarship
student_test_scholarship <- data.frame(student_names,test_scores,scholarship)
# View(student_test_scholarship) by myself


#8.Naming the columns of the dataframe to make column titles informative
names(student_test_scholarship) <- c("Student_Names","Test_Scores","Scholarship")
# View(student_test_scholarship) by myself
```

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

Answer: This data frame combines three vectors of equal length but different modes, which means it includes both numeric and character data types. However, for a matrix, its columns should have not only equal length but also the same mode. For example, the data type in a matrix should be all numeric or all character, which will make matrix multiplication and linear algegbra operations easier.

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 and name it "If_Else_Function"
If_Else_Function <- function(x) {
  if(x > 50) {
    "Pass"
  }
  else {
    "Fail"
  }
}
#11. Create a function using ifelse() and name it "Ifelse_Function"
Ifelse_Function <- function(x){
  ifelse(x > 50, "Pass", "Fail")
}

#12a. Run the first function with the value 52.5; check the result is "Pass"
Run_If_Else <- If_Else_Function(52.5)
Run_If_Else
```

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

```
#12b. Run the second function with the value 52.5; check the result is "Pass"
Run_Ifelse <- Ifelse_Function (52.5)
Run_Ifelse
```

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

```
#13a. Run the first function with the vector of test scores; this function does not work and shows that
# Run_If_Else <- If_Else_Function (test_scores)
# Run_If_Else

#13b. Run the second function with the vector of test scores; this function works and shows that "Pass"
Run_Ifelse <- Ifelse_Function (test_scores)
Run_Ifelse
```

```
## [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. According to the "Help" guide, "ifelse" functions work on all items in a vector of the same length and attributes. This "ifelse" function is vectorized, which means the function can operate all items in a vector at one time. However, the "if...else" functions only works on a length-one logical vector. There are more than one item in the vector test_scores, so the "if...else" function does not work. That is also why there is an error message saying that "Error in if (x > 50) { : the condition has length > 1" Source: R's Help Guide https://docs.ycrc.yale.edu/r-novice-gapminder/09-vectorization/

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!)