Assignment 2: Coding Basics

Christopher Starr

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.

```r
#1. Generate a sequence of numbers from one to 55, increasing by fives.
#Assign this sequence a name.

seq1.55.5.assignment01 <- seq(1,55,5) #naming the sequence
#"seq1.55.5.assignment01" then generating the sequence


#2. Compute the mean and median of this sequence.

meanseq1.55.5.assignment01 <- mean(seq1.55.5.assignment01)
#naming the mean of "seq1..."

medianseq1.55.5.assignment01 <- median(seq1.55.5.assignment01)
#naming the median of "seq1..."


#3. Ask R to determine whether the mean is greater than the median.

meanseq1.55.5.assignment01 > medianseq1.55.5.assignment01
```

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

```
#asking R to evaluate the statement Mean is greater than Median of "seq1..."
```

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.

```
#Create three vectors. Label each w/ a data type.

studentnames <- c("Tom","Mick", "Harry", "Angie")
#create student names vector - string data type.

testscores <- c(75,79,68,97) #create a list of test scores - number data type.

scholarshipstatus <- c(TRUE,FALSE,FALSE,TRUE)
#designation of scholarship status TRUE=on a scholarship - logic data type.

#Combine vectors into a data frame.
studentrecords <- data.frame(studentnames, testscores, scholarshipstatus)
#combine each independent vector into one data frame, allowing the points
#across vectors to be aligned with the name of the students.
```

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

   Answer: This data frame has different kinds of data. Student names are strings, test scores are numbers, scholarship T/F is logic (or maybe could be a 1 or 0 or even string). A matrix cannot mix data types.

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
evaluatingscoreif...else <- function(x){
  if (x>50) print("Pass") else
  print("Fail") #creating the recipe for if/else version
}


#11. Create a function using ifelse()
evaluatingscoreifelse <- function(x){
  ifelse(x>50, "Pass", "Fail")
} #creating the recipe for ifelse() version


#12a. Run the first function with the value 52.5
simplefunctionif...else <- evaluatingscoreif...else(52.5)
```

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

```
#writing instruction for asking evaluatingscoreif...else function to run with
#the score 52.5 under the name simplefunctionif...else

simplefunctionif...else #telling the function to run with 52.5 added for x.
```

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

```
#12b. Run the second function with the value 52.5
simplefunctionifelse <- evaluatingscoreifelse(52.5)
#writing instructions for asking function evaluatingscoreifelse to run with 52.5

simplefunctionifelse #telling the function to run with 52.5
```

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

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

###commenting out as this code throws an error and won't allow knit ###
#simplefunctionif.else_testscore_list <- evaluatingscoreif...else(testscores)
#writing instruction for asking evaluatingscoreif...else function to run with
#the scores in vector testscores under the name simplefunctionif...else

###commenting out as this code throws an error and won't allow knit ###
#simplefunctionif.else_testscore_list #telling the function to run with
#testscores added for x.


#13b. Run the second function with the vector of test scores
simplefunctionifelse_testscore_list <- evaluatingscoreifelse(testscores)
#writing instructions for asking function evaluatingscoreifelse to run with
#testscores inserted for x

simplefunctionifelse_testscore_list #telling the function to run with
```

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

```
#testscores for x
```

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

    > Answer: ifelse() worked and if/else did not. it seems if/else does not loop itself, or it can’t run multiple times the function automatically. it says it can only do it once: “Error in if (x > 50) print(”Pass”) else print(“Fail”) :the condition has length > 1”
    >
    > it seems ifelse() is able to vectorize or run all values from the list as a sort of loop function.

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!)