Assignment 2: Coding Basics

Asreeta Ushasri

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.First, I created a sequence of numbers from 1 to 55, at an interval of 5.

#I assigned this object the name A2.Part1.Sequence using the <- function.

seq(1,55,5)
```

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

```
A2.Part1.Sequence<-seq(1,55,5)

#2.I calculated the mean and the median using the existing functions in R.

#The mean and the median are both 26, which means this is symmetric.
#I also assigned these objects names.

mean(A2.Part1.Sequence)
```

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

```
A2.Mean.Part1.Sequence<-mean(A2.Part1.Sequence)
median(A2.Part1.Sequence)
```

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

```
A2.Median.Part1.Sequence<-median(A2.Part1.Sequence)

#3.I created an ifelse statement to examine the sequence mean and median.

#If the mean is greater than the median, the statement will return "True."
#If the mean is not greater than the median, the stateemnt will return
"False."

A2.CenterofSequence.Part1 <- function(A2.Mean.Part1.Sequence,
A2.Median.Part1.Sequence)

  ifelse(A2.Mean.Part1.Sequence>A2.Median.Part1.Sequence, print("True"),
print("False"))

A2.CenterofSequence.Part1(A2.Mean.Part1.Sequence, A2.Median.Part1.Sequence)
```

```
## [1] "False"
```

```
## [1] "False"
```

```
#4. Please reference the comments within the code for the methodology.

#I also created a function to describe the shape and skew of the
distribution.
#If the mean is greater than the median, the distribution is skewed right.
#If the mean is less than the median, the distribution is skewed left.
#If the mean and the median are identical, then the distribution is
symmetric.

A2.DistributionofSequence.Part1 <- function(A2.Mean.Part1.Sequence,
A2.Median.Part1.Sequence)

  if({A2.Mean.Part1.Sequence>A2.Median.Part1.Sequence})
{print("SkewedRight")}

  if({A2.Mean.Part1.Sequence>A2.Median.Part1.Sequence})
{print("SkewedLeft")}

  if({A2.Mean.Part1.Sequence==A2.Median.Part1.Sequence})
{print("Symmetrical")}
```

```
## [1] "Symmetrical"
```

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.First, I assigned each student name to a number in order, 1-4.

#I created four vectors. The first vector corresponds to the order 1-4.
#The next vectors include student names, test scores, and scholarship status.

#6.This is the object assignment of each student.

CeCe <- 1
Jess <- 2
Winston <- 3
Nick <- 4

#This is the number order of each student.

A2.Part2.Number<-c("1","2","3","4")

#The StudentName vector corresponds to student name.

StudentName<-c("CeCe","Jess","Winston","Nick")

#The TestScore vector lists each student's test score, in their respective
order.

TestScore<-c(94,97,98,90)

#The ScholarshipStatus lists True if a student is receiving a scholarship.
#This vector lists false if a student is not receiving a scholarship.

ScholarshipStatus<-c("True","True","True","False")

#7.Using the Help reference section in RStudio, I used the data.frame
operation.

#I combined the vectors into a data frame named A2.Part2.StudentAcademics.
#This data frame provides information about each student's academic
performance.

A2.Part2.StudentAcademics<-data.frame(StudentName, TestScore,
```

```
ScholarshipStatus)

A2.Part2.StudentAcademics

##   StudentName TestScore ScholarshipStatus
## 1        CeCe        94              True
## 2        Jess        97              True
## 3     Winston        98              True
## 4        Nick        90             False
```

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

   Answer: In a data frame, each vector must be the same length. Looking at the data frame above, each student needs a value for their test score and scholarship status in order to combine the vectors into a data frame. As discussed in Module 1 of our Data Exploration course, a matrix must have the same type of data in each column, such as all integer values. A data frame has more flexibility in combining names and integers into the same table. In this example, the data frame has both student names (strings of characters) and test scores (integers) in the same table.

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

#I created this conditional function as A2.Part3.StudentPassingGrade.
#This function uses two separate if operations.

A2.Part3.StudentPassingGrade<-function(x)
  {
  if({x>50})  {print("Pass")}
  if({x<=50})  {print("Fail")}
  }


#11.Create a function using ifelse()

#The ifelse operation combines multiple if statements into one line of code.
#I named the ifelse function as A2.Part3.StudentPassFail.
#This has a similar output to the previous function for single values.
```

```
A2.Part3.StudentPassFail<-function(y)

{ifelse(y>50, print("Pass"), print("Fail"))}


#12a.Run the first function with the value 52.5

A2.Part3.StudentPassingGrade(52.5)

## [1] "Pass"

#12b.Run the second function with the value 52.5

A2.Part3.StudentPassFail(52.5)

## [1] "Pass"

## [1] "Pass"

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

#The first function cannot process a vector through two if conditions.

#13b.Run the second function with the vector of test scores

A2.Part3.StudentPassFail(TestScore)

## [1] "Pass"

## [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 operation 'ifelse' is able to calculate the pass/fail status of the TestScore vector. According to the Yale Center for Research Computing, R vectorization allows a segment of code to operate on each value of a vector at once (YCRC, 2016). The ifelse function uses vectorization to apply the conditional statement to each value, without requiring a separate code segment. However, the 'if' ... 'else' operation requires a loop to understand how to operate on each individual value in a vector. Therefore, the ifelse function is more useful when working with vectors or multiple data points.

References

Yale Center for Research Computing. (2016). *R for Novices: Vectorization.* https://docs.ycrc.yale.edu/r-novice-gapminder/09-vectorization/

Thomas Wright and Naupaka Zimmerman (eds): "*Software Carpentry: R for Reproducible Scientific Analysis."* Version 2016.06, June 2016, https://github.com/swcarpentry/r-novice-gapminder, 10.5281/zenodo.57520.