Assignment 2: Coding Basics

Kamil Burak Karayel

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.
seq1to55by5 <- seq(1, 55, 5) # create new sequence from 1 to 55, increment by 5
seq1to55by5
```

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

```
#2.
mean_seq1to55by5 <- mean(seq1to55by5) # calculate mean of this new sequence
mean_seq1to55by5
```

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

```
median_seq1to55by5 <- median(seq1to55by5) # calculate the median of this new sequence
median_seq1to55by5
```

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

```
#3.
results <- c("mean>median","mean=median","mean<median") # create a vector to call the result from possi
result <- if(mean_seq1to55by5 > median_seq1to55by5) { #if mean>median then assign "mean>median" to 'res
  results[1]
  } else if(mean_seq1to55by5 < median_seq1to55by5) { #if mean<median then assign "mean<median" to 'resu
  results[3]
  } else {results[2]} #if mean=median then assign "mean=median" to 'result' object
result
```

```
## [1] "mean=median"
```

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
vectornames <- c("studentAAA", "studentBBB", "studentCCC", "studentDDD") #type=character
vectorscores <- c(45,55,65,75) #type=numeric
vectorscholarship <- c(TRUE,FALSE,FALSE,TRUE) #type=logical
#6
#Labeled in #5
#7 option1
df_namescorescholarship <- data.frame("names"=vectornames,"scores"=vectorscores,"scholarship"=vectorsch
df_namescorescholarship
```

```
##        names scores scholarship
## 1 studentAAA     45        TRUE
## 2 studentBBB     55       FALSE
## 3 studentCCC     65       FALSE
## 4 studentDDD     75        TRUE
```

```
#7 option2
df_name <- as.data.frame(vectornames)
df_score <- as.data.frame(vectorscores)
df_scholar <- as.data.frame(vectorscholarship)
df_opt2 <- cbind(df_name,df_score,df_scholar)
class(df_opt2)
```

```
## [1] "data.frame"
```

```
df_opt2
```

```
##   vectornames vectorscores vectorscholarship
## 1  studentAAA           45              TRUE
## 2  studentBBB           55             FALSE
## 3  studentCCC           65             FALSE
## 4  studentDDD           75              TRUE
```

```
#8
#Labeled as names, scores, scholarship
```

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

   Answer: Matrix includes same type of elements, Data Frame includes different vector 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
passXfail <- function(x) { #create a new function named passXfail
  if (x>50) examresult <- "Pass" #if the entry in the function is greater than 50 then assign "Pass" to
  else examresult <- "Fail" #otherwise assign "Fail" to examresult object
  return(examresult)
}
#11. Create a function using ifelse()
passXfail_opt2 <- function(x) { #create a new function named passXfail_opt2 as a second option
  examresult <- ifelse(x>50, "Pass", "Fail") #if the entry in the function is greater than 50 then assi
  return(examresult)
}
#12a. Run the first function with the value 52.5
passXfail(52.5) #Results "Pass" as expected.
```

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

```
#12b. Run the second function with the value 52.5
passXfail_opt2(52.5) #Results "Pass" as expected.
```

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

```
#13a. Run the first function with the vector of test scores
#> passXfail(vectorscores)
#I got an error because the condition (vectorscores) has length > 1)
#13b. Run the second function with the vector of test scores
passXfail_opt2(vectorscores) #It worked! I got "Fail" "Pass" "Pass" "Pass"
```

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

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

    Answer: 'if'...'else' function checks only one condition and can't work properly with vectors. However 'ifelse' is compatible with vectors and checks every element in the vector and results a vector again.

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!)