Assignment 2: Coding Basics

Aadya Shukla

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.
7. Initial here to acknowledge that you did not use AI at all in completing this assignment: AS

Basics, Part 1

1. Use R's `seq()` function to create a sequence of numbers from 100 to 333, increasing by threes. Assign this sequence a variable name.
2. Compute the *mean* of this sequence, assigning this values its own variable name.
3. Compute the *standard deviation* (`sd()`) of this sequence, assigning this values its own variable name.
4. Display the the mean minus the standard deviation as well as the mean plus the standard deviation.
5. Insert comments in your code to describe what you are doing.

```
#1. Using seq(from 100, to 333, by 3)
sequence <- seq(100,333,3)
sequence
```

```
##  [1] 100 103 106 109 112 115 118 121 124 127 130 133 136 139 142 145 148 151 154
## [20] 157 160 163 166 169 172 175 178 181 184 187 190 193 196 199 202 205 208 211
## [39] 214 217 220 223 226 229 232 235 238 241 244 247 250 253 256 259 262 265 268
## [58] 271 274 277 280 283 286 289 292 295 298 301 304 307 310 313 316 319 322 325
## [77] 328 331
```

```r
#2. Using command mean for the vector "sequence" created before to calculate the average
mean <- mean(sequence)
mean #Received 215.5
```

```
## [1] 215.5
```

```r
#3. Using command sd for the vector "sequence" created before to calculate the standard deviation
stdev <- sd(sequence)
stdev #Received 67.98162
```

```
## [1] 67.98162
```

```r
#4. Created 2 interval labels, one with stdev added, one with stdev subtracted
interval1 <- mean + stdev
interval2 <- mean - stdev
interval1 #Received 283.4816
```

```
## [1] 283.4816
```

```r
interval2 #Received 147.5184
```

```
## [1] 147.5184
```

Basics, Part 2

6. 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).
7. Label each vector with a comment on what type of vector it is.
8. Combine each of the vectors into a data frame. Assign the data frame an informative name.
9. Label the columns of your data frame with informative titles.

```r
#6.
Students <- c("Miriam","Jared","Aaron","Tammy")
Students
```

```
## [1] "Miriam" "Jared"  "Aaron"  "Tammy"
```

```r
#This is a character vector

TestResults <- c(73,94,81,35)
TestResults
```

```
## [1] 73 94 81 35
```

```r
#This is a numeric vector

Scholarship <- c(FALSE,TRUE,TRUE,FALSE)
Scholarship
```

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

```r
#This is a logical vector

#8.
ScholarshipResults <- data.frame(Students,TestResults,Scholarship)

#9.
names(ScholarshipResults) <- c("Student Names","Test Scores", "Scholarship Eligibility")
```

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

    Answer: This data frame has three different vectors, each of which has a different type, i.e. logical, numerical, and character. A matrix must have data of the same type while data frames can store different data types.

Basics, Part 3

11. 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, it returns the word "Pass"; otherwise it returns the word "Fail".
12. Create a second function that does the exact same thing as the previous one but uses `ifelse()` instead if `if...else`.
13. Run both functions using the value 54 as the input
14. Run both functions using the **vector** of student test scores you created as the input. (Only one will work properly...)

```r
#11. Create a function using if...else
Method1 <- function(x) {
  if(x > 50) {
    "Pass"
  }
  else {
    "Fail"
  }
}
#12. Create a function using ifelse()
Method2 <- function(x){
  ifelse(x>50, "Pass", "Fail")
}
#13a. Run the first function with the value 54
Method1(54) #Received "Pass"
```

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

```
#13b. Run the second function with the value 54
Method2(54) #Received "Pass"
```

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

```
#14a. Run the first function with the vector of test scores
#Method1(TestResults) #Received Error: condition has length >1

#14b. Run the second function with the vector of test scores
Method2(TestResults) #Received Pass,Pass,Pass,Fail
```

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

15. QUESTION: Which option of `if...else` vs. `ifelse` worked? Why? (Hint: search the web for "R vectorization"; it's ok here if an AI response is presented in the search response.)

    Answer: The second option of "ifelse" worked but "if...else" reverted back an error saying "condition has length >1". ifelse" is a vectorized function, meaning it can go through individual components of a vector and revert back with a result for each. However, "if...else" is not a vectorized function; it only works with a singular value. Since the vector "TestResults" has more than 1 component, only vectorized functions will work on it.

NOTE Before knitting, you'll need to comment out the call to the function in Q14 that does not work. (A document can't knit if the code it contains causes an error!)