Assignment 2: Coding Basics

Lexi Nelson

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: AN

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.
sequence <- seq(100,333,3) #the numbers listed in parenthesis are first, last, and count by
print(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.
mean_sequence <- mean(sequence) #R has a built in function for mean
print(mean_sequence)
```

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

```r
#3.
standard_deviation <- sd(sequence) #use <- to assign a variable name
print(standard_deviation)
```

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

```r
#4.
mean_minus_sd <- mean_sequence - standard_deviation
print(mean_minus_sd) #can use the variable names we have already assigned to perform computations
```

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

```r
mean_plus_sd <- mean_sequence + standard_deviation
print(mean_plus_sd) #use print so that R displays results
```

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

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
student_names <- c("A", "E", "C", "D") #character vector
test_scores <- c(60, 70, 80, 90) #numeric vector (integer vector)
scholarship <- c(TRUE, FALSE, FALSE, TRUE) #logical vector

test_results <- as.data.frame(student_names) #create a data frame. test_results is my informative name
class(test_results) #check this is a data frame
```

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

```r
test_results <- cbind(student_names, test_scores, scholarship)
print(test_results)
```

```
##      student_names test_scores scholarship
## [1,] "A"           "60"        "TRUE"
## [2,] "E"           "70"        "FALSE"
## [3,] "C"           "80"        "FALSE"
## [4,] "D"           "90"        "TRUE"
```

```
colnames(test_results) #shows that the vector names became the column names, and these were already inf
```

```
## [1] "student_names" "test_scores"   "scholarship"
```

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

    Answer:This data frame is different from a matrix because a matrix can only store data of a uniform data type. Here we have used our data frame to store both strings and numbers.

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...)

```
#11. Create a function using if...else
pass_or_fail <- function(x){
    if (x > 50) {
  print("Pass") #returns the word pass if x is greater than 50
} else {
  print("Fail") #returns the word fail if x is not greater than 50
}
}

#12. Create a function using ifelse()
pass_or_fail_2 <- function(x){
    ifelse (x > 50, "Pass", "Fail")
  #the second argument determines the output if the expression evaluated with the first argument is tru
}


#13a. Run the first function with the value 54
pass_or_fail(54) #returns Pass which indicates the function is working correctly
```

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

```
#13b. Run the second function with the value 54
pass_or_fail_2(54) #returns Pass which again indicates the function is working correctly
```

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

```
#14a. Run the first function with the vector of test scores
#pass_or_fail(test_scores)
#this is commented out since it generated an error message

#14b. Run the second function with the vector of test scores
pass_or_fail_2(test_scores)
```

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

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: Ifelse works but not if…else because ifelse is a vectorized function in R, meaning it is designed to operate on one element after another across the length of a vector. On the other hand, if…else is not vectorized and is only intended to function with one single input.

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!)