Assignment 2: Coding Basics

Sky Volz

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: SV

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.

```r
#1.
long_sequence <- seq (100, 333, 3) #creating a sequence from 100 to 333 by 3

#2.
mean_sequence <- mean(long_sequence) #mean of the sequence

#3.
sd_sequence <- sd(long_sequence) #standard deviation of the sequence

#4.
difference <- mean_sequence - sd_sequence # mean minus sd
add_sd_mean <- mean_sequence + sd_sequence # mean plus sd
```

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
#7
vector_names <- c("John", "Jack", "James", "Jessica")
class(vector_names) #vector type: character
```

```
## [1] "character"
```

```r
vector_scores <- c(98, 65, 73, 100)
class(vector_scores) #vector type: numeric
```

```
## [1] "numeric"
```

```r
vector_scholarship <- c(TRUE, FALSE, TRUE, FALSE)
class(vector_scholarship) #vector type: logical
```

```
## [1] "logical"
```

```r
#8 and #9
students_df <- data.frame("Names" = vector_names, "Scores" = vector_scores,
                          "Scholarship" = vector_scholarship)
print(students_df)
```

```
##     Names Scores Scholarship
## 1    John     98        TRUE
## 2    Jack     65       FALSE
## 3   James     73        TRUE
## 4 Jessica    100       FALSE
```

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

    Answer: Both a data frame and a matrix are two-dimensional structures, but a matrix can only contain elements of the same type. This is different from a data frame because in a data frame each column can have different types of elements, for example students_df has character, numeric, and logical elements.

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

if_function <- function( x ) {
  if(x > 50) {
    return("Pass")
  } else {
    return("Fail")
  }
}

#12. Create a function using ifelse()
ifelse_function <- function (x) {
  ifelse(x>50, "Pass", "Fail")
}

#13a. Run the first function with the value 54

if_function(54)
```

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

```r
#13b. Run the second function with the value 54

ifelse_function(54)
```

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

```r
#14a. Run the first function with the vector of test scores

#if_function(vector_scores)

#14b. Run the second function with the vector of test scores

ifelse_function(vector_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: The ifelse worked, and the if…else did not work. When I tried to run the if…else it gave an error saying that the condition was too long because it was a vector. Based on this, if…else did not work becaue it only works for a single condition, but the ifelse worked because it has been vectorized which means it is designed to look at entire vector of conditions.

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!)