Assignment 2: Coding Basics

Britney Pepper

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on coding basics.

Directions

1. Change "Student Name" on line 3 (above) with your name.
2. Work through the steps, **creating code and output** that fulfill each instruction.
3. Be sure to **answer the questions** in this assignment document.
4. When you have completed the assignment, **Knit** the text and code into a single PDF file.
5. After Knitting, submit the completed exercise (PDF file) to the dropbox in Sakai. Add your first and last name into the file name (e.g., "FirstLast_A02_CodingBasics.Rmd") prior to submission.

Basics Day 1

1. Generate a sequence of numbers from one to 100, increasing by fours. 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.
one_hundred_sequence <- seq(1, 100, 4) # from, to, by

#2.
mean_sequence <- mean(one_hundred_sequence)
#assigning name to the mean function and using the mean function on the sequence from part 1
mean_sequence
```

```
## [1] 49
```

```
#printing the output from the mean of the sequence

median_sequence <- median(one_hundred_sequence)
#assigning name to the median function and using the median function on the sequence from part 1
median_sequence
```

```
## [1] 49
```

```
#printig the output from the median of the sequence

#3.
mean_sequence > median_sequence
```

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

```
#testing to see if the mean is greater than th median of the sequence
#if mean is greater than median == TRUE and else it is FALSE
#The correct answer is that it is FALSE because the mean and median have the same value
```

Basics Day 2

5. Create a series of vectors, each with four components, consisting of (a) names of students, (b) test scores out of a total 100 points, and (c) whether or not they have passed the test (TRUE or FALSE) with a passing grade of 50.
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., 6., 8.
Names <- c("Sally", "Mark", "Robert", "Cole", "Margret") #vector type: characters
Grades <- c(92, 67, 88, 45, 79) #vector type: numbers
Passed <- c("TRUE", "TRUE", "TRUE", "FALSE", "TRUE") #vector type: characters

#7.
Class_Test_Scores <- data.frame(Names, Grades, Passed)
print(Class_Test_Scores)
```

```
##     Names Grades Passed
## 1   Sally     92   TRUE
## 2    Mark     67   TRUE
## 3  Robert     88   TRUE
## 4    Cole     45  FALSE
## 5 Margret     79   TRUE
```

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

   Answer: The data frame is different from a matrix because in a matrix, every value in the rows and columns are the same type of data. In this data frame, we have used different types of vectors to create colums that contain different types of data. Some are characters and others are numbers.

10. Create a function with an if/else statement. Your function should determine whether a test score is a passing grade of 50 or above (TRUE or FALSE). You will need to choose either the `if` and `else` statements or the `ifelse` statement. Hint: Use `print`, not `return`. The name of your function should be informative.
11. Apply your function to the vector with test scores that you created in number 5.

```r
#10.
determine_if_passed_test <- function(x){
  if(x > 49) {
    TRUE
    }
  else {
    FALSE
  }
}
#NOT WORKING

determine_if_passed_test_ifelse <- function(x) {
  ifelse(x > 49, "Pass", "Fail")
}

#11.
determine_if_passed_test(Grades)
```

```
## Warning in if (x > 49) {: the condition has length > 1 and only the first
## element will be used
```

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

```r
print(determine_if_passed_test(Grades))#Did not work
```

```
## Warning in if (x > 49) {: the condition has length > 1 and only the first
## element will be used
```

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

```r
determine_if_passed_test_ifelse(Grades) #Worked correctly
```

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

```r
print(determine_if_passed_test_ifelse(Grades))
```

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

12. QUESTION: Which option of `if` and `else` vs. `ifelse` worked? Why?

    Answer: The option of 'ifelse' worked. This is likely because the 'ifelse' function can run through all the values in an entire vector. The 'if' and 'else' function could only read the first the first value in the vector. Therefore, the 'if' and 'else' likely can only be used for single values instead of a vector.