Assignment 2: Coding Basics

Jake Whisler

OVERVIEW

This exercise accompanies the lessons 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 Sakai.

Basics, Part 1

1. Generate a sequence of numbers from one to 30, increasing by threes. 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. Creating a sequence from one to thirty, by increments of three,
# assigned with the name threeSequence
three_sequence <- seq(1, 30, 3)

#2. Using mean() and median() to find the mean and median of this new vector.
# Both values show in the console as 14.5.
mean(three_sequence)
```

```
## [1] 14.5
```

```
median(three_sequence)
```

```
## [1] 14.5
```

```
#3. Using the > function to determine if the mean is greater than the median.
#The console returns the value "FALSE".
mean(three_sequence) > median(three_sequence)
```

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

Basics, Part 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.

```
# Creating three vectors: student_name, test_score, passed_test

student_name <- c("Fred", "Ted", "Ned", "Bread")
# student_name vector is a character vector

test_score <- seq(25, 100, 25)
# test_score vector is a numerical vector

passed_test <- test_score >= 50
# passed_test vector is a logical vector

# Creating a data_frame from these three vectors under the name student_test_scores,
# with descriptive titles for each column
student_test_scores <- data.frame("Student Name"=student_name, "Test Score"=test_score,
                                  "Test Passed"=passed_test)
```

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

   Answer: A matrix can only contain a single type of data (numerical, categorical, etc.), while this data frame contains multiple different data types.

10. Create a function with an if/else statement. Your function should take a **vector** of test scores and print (not return) whether a given 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.

11. Apply your function to the vector with test scores that you created in number 5.

```
# Creating a function with the "ifelse" statement
are_grades_passing <- function(x) {
  ifelse(x >= 50, print("PASS"), print("FAIL"))
}

# Testing "ifelse" function with test_score vector
are_grades_passing(test_score)
```

```
## [1] "PASS"
## [1] "FAIL"
```

```
## [1] "FAIL" "PASS" "PASS" "PASS"
```

```r
# Creating a function with "if" and "else" statements
# are_grades_passing2 <- function(x) {
#  if(x >= 50) {
#    print("PASS")
#  }
#  else {
#    print("FAIL")
#  }
# }

# Testing "if" and "else" statements with test_score vector
# are_grades_passing2(test_score)

# NOTE: it would not allow me to knit the code with the "if" and "else" statements
# being non-functional, so I added the pound symbols to render them unusable.
```

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

    Answer: The “if” and “else” function did not work, as it says that the condition has a length > 1. The “ifelse” function did work, however. This is because I inserted a vector, instead of a singular value, as the “x” value, which the “if” statement seems to have conflict with. The “ifelse” function does not seem to have this issue.