Assignment 2: Coding Basics

Hannah Wudke

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.

Basics, Part 1

1. Generate a sequence of numbers from one to 55, increasing by fives. 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_to_fiftyfive_by_five_sequence <- seq(1, 55, 5)
one_to_fiftyfive_by_five_sequence
```

```
##  [1]  1  6 11 16 21 26 31 36 41 46 51
```

```
# Here, I used seq(from, to, by), to create a sequence from 1 to 55, in fives

#2.
mean(one_to_fiftyfive_by_five_sequence)
```

```
## [1] 26
```

```
median(one_to_fiftyfive_by_five_sequence)
```

```
## [1] 26
```

```
# here, I used the basic commands mean(one_to_fiftyfive_by_five_sequence) and
# median(one_to_fiftyfive_by_five_sequence) to calculate the mean and median of
# my sequence, which I previously named one_to_fiftyfive_by_five_sequence.

#3.
mean(one_to_fiftyfive_by_five_sequence) > median(one_to_fiftyfive_by_five_sequence)
```

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

```
median(one_to_fiftyfive_by_five_sequence) > mean(one_to_fiftyfive_by_five_sequence)
```

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

```
# Here, I typed both possible outcomes. Either the mean is greater than the
# median of my sequence, or the median is greater than the mean.
# Because R shows the outcomes of these conditional statements,
# I can easily see which is greater.
# Both statements were false, so the mean and median must be equal.
```

Basics, Part 2

5. 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).
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 and 6.
vector_student_names <- c("Adam", "Brian", "Carter", "David")
# vector type = character
vector_test_scores <- c(90, 75, 80, 50)
# vector type = double
vector_scholarship_status <- c(TRUE, TRUE, FALSE, FALSE)
# vector type = logical

#7.
Student_Data_Frame <- data.frame(vector_student_names,
                                 vector_test_scores, vector_scholarship_status)

#8.
names(Student_Data_Frame) <- c("Student Names", "Test Scores", "Scholarship Status")
```

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

   Answer: Although they arguably look the same, they are different because a matrix can only include a singular data type, whereas a data frame can contain multiple. For example, a data table can contain numerical data, as well as true/false statements. If a matrix contained numerical data, it could only contain numerical data.

10. 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, print the word "Pass"; otherwise print the word "Fail".

11. Create a second function that does the exact same thing as the previous one but uses `ifelse()` instead if `if...else`.

12. Run both functions using the value 52.5 as the input

13. Run both functions using the **vector** of student test scores you created as the input. (Only one will work properly...)

```
#10. Create a function using if...else
x <- vector_test_scores
pass_fail_function <- function(x)
  if (x>50) {print("PASS")} else if (x<50) {print("FAIL")}

#11. Create a function using ifelse()
x <- vector_test_scores
pass_fail_function_ifelse <- function(x)
  ifelse(x>50, "PASS", "FAIL")

#12a. Run the first function with the value 52.5

pass_fail_function(52.5)
```

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

```
#12b. Run the second function with the value 52.5

pass_fail_function_ifelse(52.5)
```

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

```
#13a. Run the first function with the vector of test scores

#this comment is to call out that this function does not work:
# pass_fail_function(vector_test_scores)

#13b. Run the second function with the vector of test scores

pass_fail_function_ifelse(vector_test_scores)
```

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

14. QUESTION: Which option of `if...else` vs. `ifelse` worked? Why? (Hint: search the web for "R vectorization")

    Answer: ifelse worked! ifelse is essentially a vectorized version of if...else. Ifelse can run through an entire vector at once, rather than each individual value in the vector.

NOTE Before knitting, you'll need to comment out the call to the function in Q13 that does not work. (A document can't knit if the code it contains causes an error!)