Assignment 2: Coding Basics

Zhiyuan Chen

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 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. I am generating a sequence here, from 1 to 100 by 4
seq(1,100,4)
```

```
##  [1]  1  5  9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
```

```
sequence1 <- seq(1,100,4)
sequence1
```

```
##  [1]  1  5  9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
```

```
#2. I am computing the mean and median of this sequence
mean(sequence1)
```

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

```
median(sequence1)
```

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

```
#3. whether the mean is greater than the median
mean(sequence1) > median(sequence1)
```

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

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.

```
# I am generating names of students
a <- c("Lucy", "Matt", "Will", "Rowan") # character vectors
a
```

```
## [1] "Lucy"  "Matt"  "Will"  "Rowan"
```

```
# I am generating test scores
b <- c(70, 40, 80, 90) # integer vectors
b
```

```
## [1] 70 40 80 90
```

```
# I am generating pass or fail the test(TRUE or False) info
c <- c(b > 50) # logical vectors
c
```

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

```
test_performance <- data.frame(a, b, c)
test_performance
```

```
##       a  b     c
## 1  Lucy 70  TRUE
## 2  Matt 40 FALSE
## 3  Will 80  TRUE
## 4 Rowan 90  TRUE
```

```
colnames(test_performance) <- c("names of students", "test scores out of 100", "pass or fail the test")
test_performance
```

```
##   names of students test scores out of 100 pass or fail the test
## 1              Lucy                     70                  TRUE
## 2              Matt                     40                 FALSE
## 3              Will                     80                  TRUE
## 4             Rowan                     90                  TRUE
```

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

   Answer: Matrices can only contain a single class of data, while data frames can consist of many different classes of data.

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.

```
passorfail <- function(x){
 ifelse(x<50, FALSE, TRUE)
}
b <- c(70, 40, 80, 90) # integer vectors
whether_pass_or_not <- passorfail(b)
whether_pass_or_not
```

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

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

    Answer: `ifelse` worked but `if` and `else` did not. `if` and `else` can only apply to the condition has length = 1, while `ifelse` can apply to the condition has length > 1. In other words, an if() statement can only check one element in a vector at one time, while by default, an ifelse() function checks each element in a vector one at a time. This allows to avoid the error I encountered earlier.