Assignment 2: Coding Basics

Siyu Dong

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.

```
Three_step_sequence <- seq(1,30,3)
Three_step_sequence
```

```
##  [1]  1  4  7 10 13 16 19 22 25 28
```

2. Compute the mean and median of this sequence.

```
mean_values <- mean(Three_step_sequence)
median_values <- median(Three_step_sequence)

mean_values
```

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

```
median_values
```

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

3. Ask R to determine whether the mean is greater than the median.

```
mean <- mean_values
median <- median_values

compare_mean_and_median <- function(mean, median){
  if (mean > median) {
    print("The mean is greater than the median.")
  }
  else if (mean < median) {
    print("The mean is smaller than the median.")
  }
  else {
    print("The mean equals to the median.")
  }
}

compare_mean_and_median(mean, median)
```

```
## [1] "The mean equals to the median."
```

4. Insert comments in your code to describe what you are doing.

```
#1. I'm generating a sequence starting from 1 and ending at 30 with a interval of 3.
# The final outcome was give the name as "Three_step_sequence". Details are as below:
Three_step_sequence <- seq(1,30,3) # Name the assigned sequence
Three_step_sequence # Print this sequence
```

```
##  [1]  1  4  7 10 13 16 19 22 25 28
```

```
#2. I'm trying to calculate the mean and median of the computed sequence.
mean_values <- mean(Three_step_sequence)
# Assign the computed sequence's mean to the variable "mean_values"
median_values <- median(Three_step_sequence)
# Assign the computed sequence's median to the variable "median_values"

mean_values # Print the result
```

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

```
median_values # Print the result
```

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

```
#3. Do the comparison between the values of the mean and the median computed from the Three_step_sequen
mean <- mean_values
# Assign the previously computed mean value to a new variable
median <- median_values
# Assign the previously computed median value to a new variable

compare_mean_and_median <- function(mean, median){
# Create the if/else function expression and divide them into 3 scenarios to discuss
```

```
  if (mean > median) {
    print("The mean is greater than the median.")
  }
  else if (mean < median) {
    print("The mean is smaller than the median.")
  }
  else {
    print("The mean equals to the median.")
  }
}

compare_mean_and_median(mean, median)
```

```
## [1] "The mean equals to the median."
```

```
# Based on the algorithm, compare the mean and the median and then print the result
```

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.

```
student_name <- c("May", "Joshua", "Bob", "Smantha")
test_scores <- c(99, 61, 45, 80)
pass_or_fail <- test_scores >= 50
```

6. Label each vector with a comment on what type of vector it is.

```
class(student_name)
```

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

```
class(test_scores)
```

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

```
class(pass_or_fail)
```

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

```
student_name # character
```

```
## [1] "May"     "Joshua"  "Bob"     "Smantha"
```

```
test_scores # numeric
```

```
## [1] 99 61 45 80
```

```
pass_or_fail #logical
```

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

7. Combine each of the vectors into a data frame. Assign the data frame an informative name.

```
student_test_info <- data.frame(student_name, test_scores, pass_or_fail)
student_test_info
```

```
##   student_name test_scores pass_or_fail
## 1          May          99         TRUE
## 2       Joshua          61         TRUE
## 3          Bob          45        FALSE
## 4      Smantha          80         TRUE
```

8. Label the columns of your data frame with informative titles.

```
colnames(student_test_info) <- c("Name", "Score", "Status")
student_test_info
```

```
##      Name Score Status
## 1     May    99   TRUE
## 2  Joshua    61   TRUE
## 3     Bob    45  FALSE
## 4 Smantha    80   TRUE
```

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

   Answer: 1. For data frame, there can be different data types among columns; for matrix, all the elements must maintain the same data type. 2. There could be column and row names respectively for a data frame, while no name can be assigned for matrix.

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.

```
#Option 1
check_scores <- function(student_scores){
  for (score in student_scores) {
    if (score >= 50) {
      print("True")
    }
    else if (score < 50) {
      print("False")
    }
  }
}

# Option 2
check_pass_or_fail <- function(student_scores){
  for (score in student_scores) {
    ifelse(score >= 50,
           print("True"),
           print("False"))
  }
}
```

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

```
#Option 1
check_scores <- function(student_scores){
  for (score in student_scores) {
    if (score >= 50) {
      print("True")
    }
    else if (score < 50) {
      print("False")
    }
  }
}

student_scores <- student_test_info$Score

check_scores(student_scores)
```

```
## [1] "True"
## [1] "True"
## [1] "False"
## [1] "True"
```

```
# Option 2
check_pass_or_fail <- function(student_scores){
  for (score in student_scores) {
    ifelse(score >= 50,
           print("True"),
           print("False"))
  }
}

student_scores <- student_test_info$Score

check_pass_or_fail(student_scores)
```

```
## [1] "True"
## [1] "True"
## [1] "False"
## [1] "True"
```

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

    Answer: They both worked!