Assignment 2: Coding Basics

Yingfan Zeng

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.

   Answer: The sequence is: 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. The code is shown below.

2. Compute the mean and median of this sequence.

   Answer: The mean of the sequence is 49, and the median is also 49.

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

   Answer: The mean is not greater than the median.

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

```
#1.
#sequence1 is a sequence of numbers from one to 100, increasing by fours
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.
#mean1 is the mean value of sequence1
mean1 <- mean(sequence1)
mean1
```

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

```
#median1 is the median value of sequence1
median1 <- median(sequence1)
median1
```

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

```
#3.
#determining if the mean is greater than the median
mean1 > median1
```

```
## [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.

```
#5.& 6.
#(a) names of students
names <- c("Tom", "Jerry", "Lucy", "Amy") #This is a character vector.

#(b) test scores out of a total 100 points
scores <- c(99, 49, 98, 47) #This is a numeric vector.

#(c) whether or not they have passed the test (TRUE or FALSE) with a passing grade of 50.
pass <- c(TRUE, FALSE, TRUE, FALSE) #This is a logical vector.

#7.
#create a data frame combining 3 vectors
Exam_Results <- data.frame(names, scores, pass)
Exam_Results
```

```
##   names scores  pass
## 1   Tom     99  TRUE
## 2 Jerry     49 FALSE
## 3  Lucy     98  TRUE
## 4   Amy     47 FALSE
```

```
#8.
#change column names of the data frame
names(Exam_Results) <- c("Names", "Scores", "Pass?")
Exam_Results
```

```
##   Names Scores Pass?
## 1   Tom     99  TRUE
## 2 Jerry     49 FALSE
## 3  Lucy     98  TRUE
## 4   Amy     47 FALSE
```

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

   Answer: A matrix is a two-dimensional data structure in which all the elements are of the same type, while a data frame can contain multiple types of data. Also, a data frame must have column and row names, while this is not necessary for a matrix.

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.

```
#create a function by "if...else" statement
Exam_result_test1 <- function(score){
  if (score >= 50){
  print(TRUE)
    }
  else {
    print(FALSE)
    }
}

#apply Exam_result_test1 function to the scores vector
Exam_result_test1(scores) #It only returns the result of the first element.
```

```
## Warning in if (score >= 50) {: the condition has length > 1 and only the first
## element will be used
```

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

```
#add a for loop to the function of Exam_result_test1 to make it work for vectors
Exam_result_test1.1 <- function(scores_vector){
for (score in scores_vector){
  if (score >= 50){
  print(TRUE)
    }
  else {
    print(FALSE)
    }
  }
}

#apply Exam_result_test1.1 function to the scores vector
Exam_result_test1.1(scores) #It works but the result is not a vector.
```

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

```r
#create a function by "ifelse" statement
Exam_result_test2 <- function(score){
  ifelse(score >= 50, TRUE, FALSE)
}

#apply Exam_result_test2 function to the scores vector
Exam_result_test2(scores) #It works.
```

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

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

    Answer: By only using either "if. . . else" or "ifelse" statement with no other functions, the option of "ifelse" worked for the vector "scores". Because only the first element of a vector would be used in the condition statement "if. . . else". "if. . . else" could work for a vector with a for loop (function Exam_result_test1.1), but the result was not a vector. However, for the ifelse() function, the input must be a logical vector and the returned value would be a vector with the same length, which makes it work for the scores vector.