Assignment 2: Coding Basics

Yin-Chia Yang

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. Generating a sequence and naming it
seq(1 ,30, 3)
```

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

```
num1 <- seq(1 ,30, 3)

#2. Computing the mean and mesian of the sequence
mean(num1)
```

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

```
mean_num1 <- mean(num1)

median(num1)
```

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

```r
median_num1 <- median(num1)

#3. Comparing the mean and the median
if (mean_num1 > median_num1){
  print ("The mean is greater than the median.")
} else {
  print("The median is greater than the mean.")
}
```

```
## [1] "The median is greater than the mean."
```

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.

```r
student_name <- c("Anna", "Bellie", "Cindy", "Diane" ) #character
test_score <- c(70, 65, 45, 55) #numeric
pass_test <- test_score >=50 #logical

df_test_result <- data.frame(
  Name = student_name,
  Test_Score = test_score,
  Passed = pass_test
)
print(df_test_result)
```

```
##     Name Test_Score Passed
## 1   Anna         70   TRUE
## 2 Bellie         65   TRUE
## 3  Cindy         45  FALSE
## 4  Diane         55   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.

```
#if, else method1
pass_fail <- function(x){
  if (x > 49){
    print(TRUE) #passed the test
}
    else{
      print(FALSE) #failed the test
    }
}
pass_fail(70)
```

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

```
pass_fail(65)
```

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

```
pass_fail(45)
```

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

```
pass_fail(55)
```

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

```
pass_fail
```

```
## function(x){
##   if (x > 49){
##     print(TRUE) #passed the test
## }
##     else{
##       print(FALSE) #failed the test
##     }
## }
## <bytecode: 0x55a1c3b54e70>
```

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

```
#if else method
pass_fail_result <- pass_fail(70)
```

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

```
pass_fail_result
```

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

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

```r
#ifelse method2

pass_fail2 <- function(x){
  ifelse(x>49, print(TRUE), print(FALSE))
}
pass_fail2(70)
```

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

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

```r
pass_fail2(65)
```

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

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

```r
pass_fail2(45)
```

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

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

```r
pass_fail2(55)
```

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

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

```r
pass_fail2
```

```
## function(x){
##   ifelse(x>49, print(TRUE), print(FALSE))
## }
## <bytecode: 0x55a1c40445b8>
```

Answer: The 'if' and 'else' method worked and the 'ifelse' didn't quite worked, it has somehow printed the outcome twice in the console. I am not certain why this is happening, but I assume it might be related to the way data is stored and access in 'inelse' is through a vector?