Assignment 2: Coding Basics

Zhenghao(Will) Lin

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.
seq_3 = seq(1, 30, by = 3)
seq_3
```

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

```
#2.
mean_seq_3 = mean(seq_3)
median_seq_3 = median(seq_3)
mean_seq_3
```

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

```
median_seq_3
```

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

```r
#3.
if(mean_seq_3 > median_seq_3){
  print("Mean of seq_3 is greater than the median")
}else{
  print("Mean of seq_3 is not greater than the median")
}
```

```
## [1] "Mean of seq_3 is not greater than the median"
```

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
#5&6
# (a) Names of students
Students <- c("Aaron", "Abby", "Adam", "Alice", "Amy")
class(Students)
```

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

```r
Students
```

```
## [1] "Aaron" "Abby"  "Adam"  "Alice" "Amy"
```

```r
#The Students vector contains the names of students who took the test, it's a charactrer vector
# (b) Test scores out of 100 points
Test_Scores <- sample(0:100,5)
class(Test_Scores)
```

```
## [1] "integer"
```

```r
Test_Scores
```

```
## [1] 80  6 72 40  0
```

```r
#The Test_Scores vector contains the scores of the students who are in the Student vector. It's an inte

# (c) Whether or not they have passed the test (TRUE or FALSE) with a passing grade of 50
Results <- Test_Scores >= 50
class(Results)
```

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

```
Results
```

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

```
#The Results vector contains the passing result of the students in Student vector. It's a logical vecto

#7&8
student_data <- data.frame(Name_of_Students = Students, Test_Score = Test_Scores, Passing_Results = Resu
student_data
```

```
##   Name_of_Students Test_Score Passing_Results
## 1            Aaron         80            TRUE
## 2             Abby          6           FALSE
## 3             Adam         72            TRUE
## 4            Alice         40           FALSE
## 5              Amy          0           FALSE
```

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

   Answer:For a matrix in R, it only reveals the values in the order that we want instead of telling you what does each row and column represent. Also Matrix is not as convenient as dataframe for reusing and reference.

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.

```
Passing_Result_1 <- function(Test_Scores){
  for(x in Test_Scores){
    if (x >= 50){
      print("True")
    }else{
      print("False")
    }
  }
}
Passing_Result_2 <- function(Test_Scores){
  ifelse(Test_Scores >= 50, "True", "False")
}
Passing_Result_1(Test_Scores)
```

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

```
Passing_Result_2(Test_Scores)
```

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

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

    Answer: Based on the results of question 11, the output of if...else... and ifelse in this case is generally the same. However, since I wrote a "for" loop for "if...else...", the format of the result is little bit different. I used the built-in "view" function in r to check the coe for "ifelse". Below is the code:

```
function (test, yes, no)
{
  if (is.atomic(test)) {
    if (typeof(test) != "logical")
      storage.mode(test) <- "logical"
    if (length(test) == 1 && is.null(attributes(test))) {
      if (is.na(test))
        return(NA)
      else if (test) {
        if (length(yes) == 1) {
          yat <- attributes(yes)
          if (is.null(yat) || (is.function(yes) && identical(names(yat),
            "srcref")))
            return(yes)
        }
      }
      else if (length(no) == 1) {
        nat <- attributes(no)
        if (is.null(nat) || (is.function(no) && identical(names(nat),
          "srcref")))
          return(no)
      }
    }
  }
  else test <- if (isS4(test))
    methods::as(test, "logical")
  else as.logical(test)
  ans <- test
  len <- length(ans)
  ypos <- which(test)
  npos <- which(!test)
  if (length(ypos) > 0L)
    ans[ypos] <- rep(yes, length.out = len)[ypos]
  if (length(npos) > 0L)
    ans[npos] <- rep(no, length.out = len)[npos]
  ans
}
```

```
## function (test, yes, no)
## {
##   if (is.atomic(test)) {
```

```
##     if (typeof(test) != "logical")
##       storage.mode(test) <- "logical"
##     if (length(test) == 1 && is.null(attributes(test))) {
##       if (is.na(test))
##         return(NA)
##       else if (test) {
##         if (length(yes) == 1) {
##           yat <- attributes(yes)
##           if (is.null(yat) || (is.function(yes) && identical(names(yat),
##             "srcref")))
##             return(yes)
##         }
##       }
##       else if (length(no) == 1) {
##         nat <- attributes(no)
##         if (is.null(nat) || (is.function(no) && identical(names(nat),
##           "srcref")))
##           return(no)
##       }
##     }
##   }
##   else test <- if (isS4(test))
##     methods::as(test, "logical")
##   else as.logical(test)
##   ans <- test
##   len <- length(ans)
##   ypos <- which(test)
##   npos <- which(!test)
##   if (length(ypos) > 0L)
##     ans[ypos] <- rep(yes, length.out = len)[ypos]
##   if (length(npos) > 0L)
##     ans[npos] <- rep(no, length.out = len)[npos]
##   ans
## }
```

Although the code seems to be way more complicated than the one that I wrote for "if...else...", at some extent, both have the same funciton.