Assignment 2: Coding Basics

Xin Zhang

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics (ENV872L) on coding basics in R.

Directions

1. Change "Student Name" on line 3 (above) with your name.
2. Use the lesson as a guide. It contains code that can be modified to complete the assignment.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document. Space for your answers is provided in this document and is indicated by the ">" character. If you need a second paragraph be sure to start the first line with ">". You should notice that the answer is highlighted in green by RStudio.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file. You will need to have the correct software installed to do this (see Software Installation Guide) Press the `Knit` button in the RStudio scripting panel. This will save the PDF output in your Assignments folder.
6. After Knitting, please submit the completed exercise (PDF file) to the dropbox in Sakai. Please add your last name into the file name (e.g., "Salk_A02_CodingBasics.pdf") prior to submission.

The completed exercise is due on Thursday, 24 January, 2019 before class begins.

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. generete a sequence from 1 to 100 step by 4, and name it "onehundred_seq"
onehundred_seq <- seq(1, 100, 4)
#2. compute the mean and median using the mean(), and median() function,
#and the results are as following:
mean(onehundred_seq)
```

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

```
median(onehundred_seq)
```

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

```
#3. use conditional statement to determine whether the mean is greater than the median
mean(onehundred_seq)>median(onehundred_seq)
```

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

> ANSWER:The consequence is generating used onehundred_seq <- seq(1, 100, 4).The mean and the median of this consequence are both 49, and as the result shows 'False',which means the mean is not greater than the median.

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.

```
#create a series of vetors
Vec_names <- c("A", "B", "C", "D") # character vector
Vec_names
```

```
## [1] "A" "B" "C" "D"
```

```
Vec_scores <- c(100, 80, 60, 40) # numeric vector
Vec_scores
```

```
## [1] 100  80  60  40
```

```
Vec_pass_status <- c(TRUE, TRUE, TRUE, FALSE) #logical vector
Vec_pass_status
```

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

```
#create a dataframe
df_student_info <- data.frame(Vec_names, Vec_scores, Vec_pass_status)
names(df_student_info) <- c("Student_Name", "Score", "Pass_OR_NOT")
View(df_student_info)
```

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

   ANSWER: In a data frame, you can have different type vectors, but in a matrix, the vetor type should be consistent.

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. 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 to determine pass or not with 'if' and 'else'
pass_or_not <- function(score) {
  i = length(score) # the total numbers of elements in the score vector
  n=1
  status<-c()
 for (n in 1:i) { # for each element in the score vector, determine it is pass or not
   if (score[n] >= 50) {
    status[n] <- TRUE
  }
  else{
    status[n] <- FALSE
  }
   n= n + 1
 }
  return(status[1:i]) # combine the pass statuse in one vector
}
```

```r
#apply the function
View(Vec_scores)
pass_status<- pass_or_not(score = Vec_scores)
pass_status
```

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

```r
#solution 2, use ifelse()
pass_or_not2 <- function(score) {
   ifelse(score >= 50, TRUE, FALSE)
}
pass_or_not2(score=Vec_scores)
```

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

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

    ANSWER: Both of them can work, but when using 'if' and 'else', a loop to define the element sequence is also needed, otherwise, only the output will only include the first element. However, when using 'ifelse', it can be direcly used without a loop.