Assignment 2: Coding Basics

Blair Johnson

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.
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(1, 100, 4)
```

```
##  [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
```

```
# first value in function is starting number
#second value is the number I end with
#third value represents the increments each number increases by
sequence1 <- seq(1, 100, 4) #Assigning name to the sequence
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.
mean(sequence1) #Mean is 49
```

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

```
mean.1 <-mean(sequence1) #Created a name for the mean of the sequence

median(sequence1) #Median is 49
```

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

```
median.1 <-median(sequence1) #Created a name for the median of the sequence

#3.
mean.1 > median.1 #Asked R whether the mean is greater than median. Answer is false.
```

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

```
#4.
#See above for comments in steps 1-3.
```

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
student.names<-c("Sarah", "Drew", "Hunter", "Amanda")

test.scores<-c(90, 45, 81, 76)

whether.pass<-c(TRUE,FALSE, TRUE, TRUE)


#6
class(student.names) #Vector is character
```

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

```
class(test.scores) #Vector is numeric
```

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

```
class(whether.pass) #Vector is logical
```

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

```
#7

student.test.scores<-data.frame(student.names, test.scores, whether.pass)#Name of data frame
student.test.scores
```

```
##   student.names test.scores whether.pass
## 1         Sarah          90         TRUE
## 2          Drew          45        FALSE
## 3        Hunter          81         TRUE
## 4        Amanda          76         TRUE
```

```
#8
colnames(student.test.scores)<- c("Student.Names", "Test.Scores","Whether.Pass" )
student.test.scores
```

```
##   Student.Names Test.Scores Whether.Pass
## 1         Sarah          90         TRUE
```

```
## 2          Drew           45        FALSE
## 3        Hunter           81         TRUE
## 4        Amanda           76         TRUE
```

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

   Answer: This data frame is different from a matrix because this data frame contains data of different types such as character (student names) and numeric types (test scores) and logical types (pass or not). In a matrix, the data types for each vector must be of the same type. Therefore, student names, test scores, and pass status could not be contained in 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.

```
x<-student.test.scores$Test.Scores #creating x, which is the test scores

Is.Score.Passing<-function(x){ifelse(x>=50, print(TRUE), print(FALSE))}
#Function using `ifelse` statement

Is.Score.Passing.2<-function(x)
  if(x>=50) {
    print(TRUE)
  } else{print(FALSE)}
#Function using `if` and `else` statements, only returns 1 value
```

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

```
Score.Passing<-Is.Score.Passing(student.test.scores$Test.Scores)
```

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

```
Score.Passing
```

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

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

    Answer: For this scenario, we use "ifelse" because it runs through all of the numbers in the vector to determine which scores are passing and which scores are not.When we use the 'ifelse' feature, we create a function within a function by assigning TRUE to the values that satisfy the function and FALSE to the values that don't satisfy the function. When we create a function using the if and else feature, it only determines whether the first value in the vector is a passing score and does not return TRUE or FALSE for the following values in the vector. In this instance, the else is only applied if none of the conditions are true (ex. vector doesn't contain any values >=50).