Assignment 2: Coding Basics

Nancy Bao

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.
seqbyfour<-seq(1,100,4) # I used the seq(start#, to#, by#) function
#this specifies a sequence of 1 to 100, counting by fours.
seqbyfour #I named the sequence "seqbyfour" and I am calling it in this line.
```

```
##  [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.
seqbyfour_mean<-mean(seqbyfour) # I used mean() of the sequence, "seqbyfour".
#I created an object for the mean and specified the name of object as seqbyfour_mean.
seqbyfour_mean #calling the mean in this line.
```

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

```
#The mean of the seqbyfour sequence is 49.
seqbyfour_median<-median(seqbyfour) # I used median() on "seqbyfour" to find the median.
#Made an object for the median and specified name of the object as seqbyfour_median.
seqbyfour_median #calling the median in this line. The median is 49 for this sequence.
```

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

```
#3.
seqbyfour_mean>seqbyfour_median
```

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

```
#I used the `>` conditional statement on the mean and median objects
#The answer returned is FALSE.
#So, the mean is not greater than the median.
#We can confirm this as the mean value equals the median value, which is 49.
```

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.

```
Student_names<-c("Robert","Eda","Emily","Zachary") #Student_names is a character vector
Test_scores_outof100<-c(98,48,85,50) #Test_scores_outof100 is a numeric vector
test_scores_pass50<-c(TRUE,FALSE,TRUE,TRUE) #test_scores_pass50 is a logical vector
#test_scores_pass50 vector shows TRUE (test score is 50 or above)
#test_scores_pass50 vector shows FALSE (test score<50)
#7.Combined the vectors into a data frame. I named the dataframe,"TestResults_df".
TestResults_df<-data.frame("Student.Name"=Student_names,
                           "Test.Score.outof100"=Test_scores_outof100,
                           "Passed.test.with.score.of.50.or.higher"=test_scores_pass50)
#I gave the columns informative titles.
TestResults_df #calling the data frame in this line
```

```
##   Student.Name Test.Score.outof100 Passed.test.with.score.of.50.or.higher
## 1       Robert                  98                                   TRUE
## 2          Eda                  48                                  FALSE
## 3        Emily                  85                                   TRUE
## 4      Zachary                  50                                   TRUE
```

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

   Answer: This data frame is different from a matrix because a matrix contains row and column vectors of the same element types (e.g only numeric or only character) where as a dataframe is more diverse in which you can combine different types of vectors into a table (e.g. you can have numeric and character types of data). This dataframe has three different types of data: character, numeric, and logical.

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.

```
#For question 10:
#test_pass_check function is used to check if test score passes or fails
# a grade of 50 or higher (out of 100) outputs as TRUE, meaning student's score passed
#below 50 outputs as FALSE, meaning student's score did not pass
test_score_pass_check<- function(x) {
 print(ifelse(x<50,"FALSE","TRUE")) # my logical expression is x<50,
#so for grades lower than 50, function prints FALSE (student failed the test)
#for anything else (50 and above),function prints TRUE (student passed the test).
```

```
}
test_score_pass_check(Test_scores_outof100)
```

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

```
#My input to function is my Test_scores_outof100 numeric vector
#In this case, only one grade was FALSE.
#To check the specific student and score that did not pass:
#I know the output was the second component (so second row) in the data frame
#Using TestResults_df[2,],learned in lab, I know Eda did not pass with score of 48.
```

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

    Answer: The ifelse option worked. When I tried doing the if and else function, I got this message in return, "the condition has length >1 and only the first element will be used," which means that the function wasn't able to run all my components in the Test_scores_outof100 vector. My vector is greater than one in length (I had 4 test results); so, I knew then that I needed to use the ifelse option. At first, I tried to input ifelse(x<50,print("FALSE"),print("TRUE")) and that resulted in an output: [1] "FALSE" [1] "TRUE" [1] "TRUE" "FALSE" "TRUE" "TRUE"
    I did not want that so I tried and nested the ifelse statement inside the print function to get the ifelse function to work so that my output was just: [1] "TRUE" "FALSE" "TRUE" "TRUE".