Assignment 2: Coding Basics

Gaby Garcia

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.

```
###Using sequence function to create a sequence of numbers
firstseq<-seq(1,100, 4)
firstseq
```

```
##  [1]  1  5  9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89
## [24] 93 97
```

2. Compute the mean and median of this sequence.

```
###mean of firstseq
mean(firstseq)
```

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

```
###Median of firstseq
median(firstseq)
```

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

3. Ask R to determine whether the mean is greater than the median.

```
###Use greater than sign to determine whether mean>median
mean(firstseq)>median(firstseq)
```

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

4. Insert comments in your code to describe what you are doing.

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.

Character Vector

```
Students<-c("Lucy", "Jimmy", "Gaby", "Austin")
```

Numeric Vector

```
Scores<-c(93, 34, 88, 51)
```

Logical Vector

```
Result<-c("TRUE", "FALSE", "TRUE", "TRUE")
```

6. Label each vector with a comment on what type of vector it is.

```
##Labeled up above
```

7. Combine each of the vectors into a data frame. Assign the data frame an informative name.

```
SchoolData<-data.frame(Students, Scores, Result)
SchoolData
```

```
##   Students Scores Result
## 1     Lucy     93   TRUE
## 2    Jimmy     34  FALSE
## 3     Gaby     88   TRUE
## 4   Austin     51   TRUE
```

8. Label the columns of your data frame with informative titles.

```
names(SchoolData)<-c("Student_Names", "Student_Scores", "Student_Results")
```

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

ANSWER: This data frame is different from a matrix because data frames are more general and are able to store different modes of data (characters, numbers, or logical). All columns in a matrix are only able to store the same data type, and must have the same length.

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.

```
TestScoreResults<-function(x){
if(x >= 50){
print("TRUE")
}

else{
print("FALSE")
}}
```

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

```
Scores<-c(93, 34, 88, 51)
Students<-c("Lucy", "Jimmy", "Gaby", "Austin")

lapply(Scores, TestScoreResults)  #first argument is the Scores vector, second argument is the function
```

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

```
## [[1]]
## [1] "TRUE"
##
## [[2]]
## [1] "FALSE"
##
## [[3]]
## [1] "TRUE"
##
## [[4]]
## [1] "TRUE"
```

Try ifelse function

```
TestScoreResults2 <- function(x){
  ifelse(x>=50, "Pass", "Fail")

}

lapply(Scores, TestScoreResults2)
```

```
## [[1]]
## [1] "Pass"
##
## [[2]]
## [1] "Fail"
##
## [[3]]
## [1] "Pass"
##
## [[4]]
## [1] "Pass"
```

Lucy

```
TestScoreResults2(93)
```

```
## [1] "Pass"
```

Jimmy

```
TestScoreResults2(34)
```

```
## [1] "Fail"
```

Gaby

```
TestScoreResults2(88)
```

```
## [1] "Pass"
```

Austin

```
TestScoreResults2(51)
```

```
## [1] "Pass"
```

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

ANSWER: Both options worked for me. The syntax of the 'if' and 'else' statement is:

if (test) {statement1} else {statement2}. If the test expression is true, then statement1 will be executed. But if the test expression is false, then statement2 will be executed. In this instance, I choose the 'if' and 'else' option because if the Boolean expression of x>=50 evaluates to be true, then the if block of code will be executed (print "TRUE"), otherwise the else block of code will be executed (print "FALSE").

The syntax of the Iifelse' statement is: ifelse(test_expression, x, y). The 'ifelse' expression worked because it requires a vector. This returned vector has element from x ("Pass") if the corresponding value of test_expression is TRUE or from y("Fail") if the corresponding value of test_expression is FALSE.