Assignment 2: Coding Basics

Laurie F Muzzy

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.

```
# generating a sequence of numbers 1 through 100, by fours
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
## [24] 93 97
```

```
# naming the object sequence
hundred_by_fours <- seq(1, 100, 4)
# computing the mean and median of the sequence
mean(seq(1, 100, 4))
```

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

```
median(seq(1, 100, 4))
```

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

```
# determining if the mean is greater than the median
mean(seq(1, 100, 4)) > median(seq(1, 100, 4))
```

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

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.

```
vectora <- c("B-dawg", "Nil-baby", "Philippe", "J-money") #names of students is character vector
vectora
```

```
## [1] "B-dawg"   "Nil-baby" "Philippe" "J-money"
```

```
vectorb <- c(77, 99, 47, 85) #test scores is a numerical vector
vectorb
```

```
## [1] 77 99 47 85
```

```
vectorc <- c(TRUE, TRUE, FALSE, TRUE) #passing = TRUE is a logical vector
vectorc
```

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

```
four_student_scores_dataframe <- data.frame(vectora, vectorb, vectorc)
names(four_student_scores_dataframe) <- c("nicknames", "scores", "passed")
```

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

   ANSWER: A matrix is a table (a 2-dimensional version of a vector), and can only have one type of data A data frame is also a table, but it can have different types of data in it (character, numerical, 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. The name of your function should be informative.
11. Apply your function to the vector with test scores that you created in number 5.

```
pass_with_50 <- function(x) {
  if(x > 50) {
    TRUE
  }
  else if(x < 50) {
    FALSE
  }
  else
    TRUE
}

pass_with_50(four_student_scores_dataframe$scores)
```

```
## Warning in if (x > 50) {: the condition has length > 1 and only the first
## element will be used
```

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

```
higherthan_50 <- function(x) {
  ifelse (x > 50, TRUE, FALSE)
```

```
}
 x = four_student_scores_dataframe
higherthan_50(four_student_scores_dataframe$scores)
```

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

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

    ANSWER: If and else didn't get us what we wanted; it only worked to the extent of the first data entry in the scores column. That function can be useful if there are more data types to process. (hopefully I learn about that too!) The ifelse function worked to show the entries we were looking for. Since the arguments are limited with ifelse, it wouldn't be useful for larger more complex questions, but it works for our little dataframe.