Assignment 2: Coding Basics

Sarah Ko

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. create sequence
seq_1to100 <- seq(1, 100, 4); seq_1to100
```

```
##  [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. calculate  mean & median
seq_mean <- mean(seq_1to100); seq_mean
```

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

```
seq_median <- median(seq_1to100); seq_median
```

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

```
#3. check is mean is greater than the median
check_mean_greater_median <- seq_mean > seq_median; check_mean_greater_median
```

```
## [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.

```
#5. create vectors
student_names <- c("Amy", "Jake", "Charles", "Rosa"); student_names # character vector
```

```
## [1] "Amy"     "Jake"    "Charles" "Rosa"
```

```
test_scores <- c(98, 46, 55, 92); test_scores # numeric vector
```

```
## [1] 98 46 55 92
```

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

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

```
#7. combine vectors into data frame
student_test_results <- data.frame(student_names, test_scores, passed_test); student_test_results
```

```
##   student_names test_scores passed_test
## 1           Amy          98        TRUE
## 2          Jake          46       FALSE
## 3       Charles          55        TRUE
## 4          Rosa          92        TRUE
```

```
#8. relabel column names
names(student_test_results) <- c("Student", "Score out of 100", "Passed?"); student_test_results
```

```
##   Student Score out of 100 Passed?
## 1     Amy               98    TRUE
## 2    Jake               46   FALSE
## 3 Charles               55    TRUE
## 4    Rosa               92    TRUE
```

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

   ANSWER: Data frames and matrices are both 2 dimensional structures, but data frames can contain columns with different modes. Matrices can contain only elements of the same type.

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.

```
#10. create function to determine if test score is a passing grade of > or =50
did_student_pass <- function(x){
  student_pass_results <- ifelse(test_scores >=50, "True", "False")
}

#11. use function on vector of test scores
test_results_w_function <- did_student_pass(test_scores); test_results_w_function
```

```
## [1] "True"  "False" "True"  "True"
```

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

    ANSWER: When creating a custom function, the 'ifelse' option worked. This is because the functions 'if' and 'else' work on single elements. To use this logic with a vector, the ifelse function should be used instead.