# Assignment 2: Coding Basics

Megan Lundequam

## 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.

# using seq command to create a sequence of numbers from 1 to 100, increasing by 4's

four_sequence <- seq(1, 100, 4)

#2.

# generating objects representing the mean and median of the four_sequence vector

mean_four_sequence <- mean(four_sequence)
mean_four_sequence
```

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

```
median_four_sequence <- median(four_sequence)
median_four_sequence
```

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

```
#3.

# using conditional statements to compare them

mean_four_sequence > median_four_sequence
```

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

```
mean_four_sequence < median_four_sequence
```

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

```
mean_four_sequence == median_four_sequence
```

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

## 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.

```
# creating vectors
student_names <- c("Annie", "Megan", "Sam", "Matt")
test_scores <- c(60,75,45,90)
pass_or_not <- c("TRUE", "TRUE", "FALSE", "TRUE")

# combining into data frame

df_student_names <- as.data.frame(student_names)
df_test_scores <- as.data.frame(test_scores)
df_pass_or_not <- as.data.frame(pass_or_not)

df_students_testscores_passfail <- cbind(df_student_names,df_test_scores,df_pass_or_not)
df_students_testscores_passfail
```

```
##   student_names test_scores pass_or_not
## 1         Annie          60        TRUE
## 2         Megan          75        TRUE
## 3           Sam          45       FALSE
## 4          Matt          90        TRUE
```

```
# renaming columns using alternative data.frame construction method
new_df_students_testscores_passfail <- data.frame("Student Name"=student_names,"Test Score"=test_scores
new_df_students_testscores_passfail
```

```
##   Student.Name Test.Score Pass.Fail
## 1        Annie         60      TRUE
## 2        Megan         75      TRUE
## 3          Sam         45     FALSE
## 4         Matt         90      TRUE
```

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

   Answer: A matrix is a series of vectors, organized into columns, where all of the columns have the same mode and length. Alternatively, this data frame is composed of columns/variables with different modes (characters and numbers) but still 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. 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.

```
# if else statement
pass.fail.test <- function(x){
  if (x >= 50) {TRUE}
  else {FALSE}
}

pass.fail.annie <- pass.fail.test(60); pass.fail.annie
```

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

```
pass.fail.megan <- pass.fail.test(75); pass.fail.megan
```

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

```
pass.fail.sam <- pass.fail.test(45); pass.fail.sam
```

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

```
pass.fail.matt <- pass.fail.test(90); pass.fail.matt
```

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

```
pass.fail.all <- pass.fail.test(test_scores); pass.fail.all
```

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

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

```
# ifelse statement

ifelse.pass.fail <- function(x){
  ifelse(x>=50,"PASS","FAIL")
}

ifelse.pass.fail.annie <- ifelse.pass.fail(60); ifelse.pass.fail.annie
```

```
## [1] "PASS"
```

```
ifelse.pass.fail.megan <- ifelse.pass.fail(75); ifelse.pass.fail.megan
```

```
## [1] "PASS"
```

```
ifelse.pass.fail.sam <- ifelse.pass.fail(45); ifelse.pass.fail.sam
```

```
## [1] "FAIL"
```

```
ifelse.pass.fail.matt <- ifelse.pass.fail(90); ifelse.pass.fail.matt
```

```
## [1] "PASS"
```

```
ifelse.pass.fail.all <- ifelse.pass.fail(test_scores); ifelse.pass.fail.all
```

```
## [1] "PASS" "PASS" "FAIL" "PASS"
```

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

Answer: For inputting the individual scores, both the “if” and “else” and “ifelse” functions worked. But when inputting the vector “test_scores”, only the “ifelse” function produced results that ran each value within the vector through the function. The “if” and “else” function did not work because in this function, if the condition has a length greater than 1, only the first element will be used.