Assignment 2: Coding Basics

Benjamin Culberson

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.

#Here I am creating the sequence using the seq() command, the 1 is the starting point,
#the 100 is the ending point, the 4 is what we're counting by

#It should be noted that this will not result in us actually counting to 100,
#we'll get to 97 and then stop

one_hundred_sequence <- seq(1,100,4)
#one_hundred_sequence

#2.

#Using the mean() command, we'll find the mean of the sequence,
#we'll do the same with the median() function

mean(one_hundred_sequence)
```

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

```r
median(one_hundred_sequence)
```

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

```r
#3.

#Here I've generated my own function that will determine if the mean is greater than the median.
#It just uses a simple ifelse statement and returns either one of two strings.
#I've also assigned the mean and median of the sequence to a value so that it looks cleaner.

mean <- mean(one_hundred_sequence)
median <- median(one_hundred_sequence)

mean_check <- function(mean, median){
  ifelse(mean > median, "True", "False")
  }

mean_check(mean, median)
```

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

```r
#Given that the mean and the and the median are the same, this function returns "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.

```r
#Name of students vector
student_name <- c("Ben", "Dave", "Brad", "Alex")
#this is a chr vector

#Test scores vector
test_scores <- c(100, 48, 21, 35)
#this is a num vector

#Pass or fail vector
pass <- c(TRUE, FALSE, FALSE, FALSE)
#this is a logi vector

df <- cbind(student_name, test_scores, pass)
colnames(df) <- c("Student Name", "Test Scores", "Pass?")
df
```

```
##      Student Name Test Scores Pass?
## [1,] "Ben"        "100"       "TRUE"
## [2,] "Dave"       "48"        "FALSE"
## [3,] "Brad"       "21"        "FALSE"
## [4,] "Alex"       "35"        "FALSE"
```

```
Test_Results.df <- as.data.frame(df)
Test_Results.df$`Test Scores` <- as.numeric(Test_Results.df$`Test Scores`)
Test_Results.df$`Pass?` <- as.logical(Test_Results.df$`Pass?`)

Test_Results.df
```

```
##   Student Name Test Scores Pass?
## 1          Ben         100  TRUE
## 2         Dave          48 FALSE
## 3         Brad          21 FALSE
## 4         Alex          35 FALSE
```

```
#I realize there is a more efficient way to create this dataframe, but this works too
```

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

   Answer: A matrix is a collection of all of the same type of thing, like "chr". A data frame, like this one, has columns of different kinds of things, "chr", "dbl", "lgl".

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.

```
Pass_Fail <- function(Score){
  ifelse(Score>=50, TRUE, FALSE)
}

Pass_Fail(test_scores)
```

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

```
Pass_Fail2 <- function(Score){
  if(Score>=50){
    print("TRUE")
  }
  else{print("FALSE")}
}

Pass_Fail2(test_scores)
```

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

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

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

    Answer: I used the "ifelse" statement and it worked just fine for the entire vector. It seems that when I used "if" and then "else", I get a message that says 'the condition has length > 1 and only the first element will be used[1] "TRUE" ' so apparantly using "if" then "else" cannot work through an entire vector, it can only work through one element at a time.