Assignment 2: Coding Basics

Courtney Horn

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.
#tinytex::reinstall_tinytex()
counting_by_four <- seq(1,100,4)
#In the code above, I am calculating a sequence of numbers from one to 100, increasing by fours. I also

#2.
mean(counting_by_four)
```

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

```
median(counting_by_four)
```

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

```
#I am calculating the mean and median of the sequence I created.

#3.
mean(counting_by_four) > median(counting_by_four)
```

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

```
#I asked R whether the mean is greater than the median.
```

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
  #a)
students_vector <- c("Courtney","Taylor","Lollie", "Bear") # character vector
  #b)
testscore_vector <- c(70,90,85,62) # numeric vector
testscore_vector
```

```
## [1] 70 90 85 62
```

```
  #c)
passed_function <- function(x){
ifelse(x>50, TRUE, FALSE) #log_exp, if TRUE, if FALSE
}
passed_function(testscore_vector)
```

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

```
passed_vector <- passed_function(testscore_vector)
passed_vector
```

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

```
#6
students_vector_char <- students_vector
class(students_vector_char)
```

```
## [1] "character"
```

```
#students_vector_char is a character vector

testscore_vector_num <- testscore_vector
class(testscore_vector_num)
```

```
## [1] "numeric"
```

```
#testscore_vector_num is a numerical vector

passed_vector_log <-passed_vector
class(passed_vector_log)
```

```
## [1] "logical"
```

```
#passed_vector_log is a logical vector



#7 and #8
testperf_df <- data.frame("student" = students_vector_char, "score"=testscore_vector,"pass/nopass"=passe
testperf_df
```

```
##    student score pass.nopass
## 1 Courtney    70        TRUE
## 2   Taylor    90        TRUE
## 3   Lollie    85        TRUE
## 4     Bear    62        TRUE
```

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

   Answer: This data frame consists of vectors of different classes (a character vector, a numerical vector, and a logical vector). Matrixes consist of vectors of the same class.

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.

```
#10
passed_function <- function(x){
ifelse(x>50, TRUE, FALSE) #log_exp, if TRUE, if FALSE
}
passed_function(40)
```

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

```
who_passed_results <- function(x){
ifelse(x>50, TRUE, FALSE) #log_exp, if TRUE, if FALSE
}

#11
who_passed_results <- function(x){
ifelse(x>50, TRUE, FALSE) #log_exp, if TRUE, if FALSE
}

who_passed_results(testscore_vector)
```

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

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

    Answer: 'Ifelse' worked. Any of these would have worked. 'Ifelse' is a way to combine 'if' and 'else', and use fewer lines of coding to accomplish the same task.