Assignment 2: Coding Basics

Matthew Vining

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on coding basics.

Directions

1. Rename this file `<FirstLast>_A02_CodingBasics.Rmd` (replacing `<FirstLast>` with your first and last name).
2. Change "Student Name" on line 3 (above) with your name.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file.
6. After Knitting, submit the completed exercise (PDF file) to Sakai.

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.
basics_sequence <- seq(1, 100, 4) ### assigning bascis_sequence a array of number 1
#through 100 and by increments of 4

#2.
mean_b <- mean(basics_sequence)
mean_b
```

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

```
# calculating and assigning the mean of the basics_sequence array

med_b <- median(basics_sequence)
med_b
```

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

```
# calculating and assigning the median of the basics_sequence array

#3.
mean_b > med_b
```

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

```
#setting the conditional statement that the mean of b is greater than the
#median of b to see if R returns a true or false statement
```

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.

```
vector_names <- c("Matthew","Nef","Abby","Sitara") # student names character vector
vector_names
```

```
## [1] "Matthew" "Nef"     "Abby"    "Sitara"
```

```
vector_testscore <- c(98,80,95,88) # numeric vector of test scores
vector_testscore
```

```
## [1] 98 80 95 88
```

```
vector_passed <- c(TRUE,TRUE,TRUE,TRUE) #logical vector if they passed test or not
vector_passed
```

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

```
test_dataframe <- data.frame(vector_names,vector_testscore,vector_passed)
test_dataframe
```

```
##   vector_names vector_testscore vector_passed
## 1      Matthew               98          TRUE
## 2          Nef               80          TRUE
## 3         Abby               95          TRUE
## 4       Sitara               88          TRUE
```

```
#create a data frame that combines each of the vectors described above

names(test_dataframe) <- c("Name","Test Score","Pass(TRUE)/Fail(FALSE)"); print(test_dataframe)
```

```
##      Name Test Score Pass(TRUE)/Fail(FALSE)
## 1 Matthew         98                   TRUE
## 2     Nef         80                   TRUE
## 3    Abby         95                   TRUE
## 4  Sitara         88                   TRUE
```

```
# labeled columns of data
```

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

   Answer: Unlike a matrix where columns have to have the same data types, columns in a data frame can have different types of data like character vs numeric.

10. Create a function with an if/else statement. Your function should take a **vector** of test scores and print (not return) whether a given 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.
11. Apply your function to the vector with test scores that you created in number 5.

```r
pass_fail <- function(vector_testscore) {
 ifelse(vector_testscore >= 50, "Pass", "Fail") #log_exp, if TRUE, if FALSE
}

pass_fail <- function(vector_testscore) {
  if(vector_testscore >= 50) {
    "Pass"
  }
  else {
    "Fail"
  }
}
TestResultMatthew <- pass_fail(98); TestResultMatthew
```

```
## [1] "Pass"
```

```r
TestResultNef <- pass_fail(80); TestResultNef
```

```
## [1] "Pass"
```

```r
TestResultAbby <- pass_fail(95); TestResultAbby
```

```
## [1] "Pass"
```

```r
TestResultSitara <- pass_fail(88); TestResultSitara
```

```
## [1] "Pass"
```

```r
# running if else test to see who passes or not
```

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

    Answer: ifelse worked the best because it was easier to write. The if and else function required a more complex loop and code to create and I found it easier to run into issues than if I use the ifelse code langauge. However, both types worked and gave the same output.