Assignment 2: Coding Basics

Sayra Martinez

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, Part 1

1. Generate a sequence of numbers from one to 30, increasing by threes. 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.
seq(1, 30, 3) #I created the sequence setting the conditions
```

```
##  [1]  1  4  7 10 13 16 19 22 25 28
```

```
kplus3_seq <- seq(1, 30, 3) ## I named the sequence.
kplus3_seq #I call up the sequence
```

```
##  [1]  1  4  7 10 13 16 19 22 25 28
```

```
#2.
mean(kplus3_seq)
```

```
## [1] 14.5
```

```
median(kplus3_seq)
```

```
## [1] 14.5
```

```
summary(kplus3_seq) # Although I used the specific computation, this function                als
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    7.75   14.50   14.50   21.25   28.00
```

```
#3.
mean(kplus3_seq) > median(kplus3_seq) #The instruction requested this,                        but
```

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

```
mean(kplus3_seq) < median(kplus3_seq)
```

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

```
mean(kplus3_seq) == median(kplus3_seq)
```

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

Basics, Part 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. I create the vectors
Student_Name <- c("Eve", "Luis", "Silvana", "Ainah") # Character type
test_scores <- c(90, 45, 99, 50) # numeric type
passed <- test_scores >= 50 & test_scores <= 100 # logical type

#6.I determine the type of data of each vector
class(Student_Name)
```

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

```
class(test_scores)
```

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

```r
class(passed)
```

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

```r
#7. I create the data frame combining the vectors, which already has the informative titles.
df_scores <- as.data.frame(cbind(Student_Name,test_scores,passed))
class(df_scores)
```

```
## [1] "data.frame"
```

```r
df_scores
```

```
##   Student_Name test_scores passed
## 1          Eve          90   TRUE
## 2         Luis          45  FALSE
## 3      Silvana          99   TRUE
## 4        Ainah          50   TRUE
```

```r
#8. They already have informative titles as showed below
colnames(df_scores)
```

```
## [1] "Student_Name" "test_scores"  "passed"
```

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

   Answer: While the matriz only allows to enter data of the same type, the data frame has different types. In this case, it contains, character, numeric and logical information.

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
#10. I create the function
approved <- function(x){
  ifelse(x>=50 & x<=100, print(TRUE), print(FALSE))
}

#11. I apply the function "approved" for the values in my vector "test_scores"
results <- approved(test_scores)
```

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

```r
results #ask difference from print(results)
```

```
## [1]  TRUE FALSE  TRUE  TRUE
## This is for my own: approved(49)
#approved(c(48, 49, 56, 100, 101))
```

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

    Answer: I used “ifelse” function, because I think it was a simple logical request, but if else, could help me to better track my conditions in more complex cases