

Assignment 2: Coding Basics

Hannah Nelson

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.  
#creating a sequence from 1-30 that counts by 3  
sequence <- seq(1, 30, by=3)  
sequence
```

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

```
#2.  
#finding the mean and median values of the sequence and naming those objects  
mean <- mean(sequence)  
mean
```

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

```
#the mean is 14.5  
median <- median(sequence)  
median
```

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

```
#the median is 14.5
```

```
#3.
```

```
#creating a function that will indicate if the mean is larger than the median by generating true/false
```

```
measures <- function(sequence) {  
  measures <- ifelse(mean > median, "TRUE", "FALSE")  
  print(measures)  
}
```

```
#creating an if else function to generate text indicating if the mean is greater, less than, or equal to
```

```
#this is useful because I know the mean and median are the same value, but that could not be reflected
```

```
measures2 <- function(sequence) {  
  measures2 <- if (mean > median) {  
    print("The mean is greater than the median")  
  }  
  else if (mean < median) {  
    print("The mean is less than the median")  
  }  
  else {  
    print("The mean and median are equal")  
  }  
}
```

```
#run sequence through functions to generate results
```

```
#this function tells us that the mean is not larger than the median
```

```
measures(sequence)
```

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

```
#this function tells us the mean is equal to the median
```

```
measures2(sequence)
```

```
## [1] "The mean and median are equal"
```

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. & #6.
```

```
#creating vector with four names called "names"
```

```
names <- c("Bella", "Emily", "Alex", "Rachel")
```

```

#creating vector with four scores called "scores"
scores <- c(80, 90, 70, 40)
#creating vector with true/false values indicating if a passing score was achieved corresponding to the
pass <- c("TRUE", "TRUE", "TRUE", "FALSE")

#7. & #8.
#creating data frame called "testing" by combining the three vectors and naming the columns
testing <- data.frame("names"=names, "scores"=scores, "passing_score"=pass)
testing

```

```

##      names scores passing_score
## 1  Bella      80           TRUE
## 2  Emily      90           TRUE
## 3   Alex      70           TRUE
## 4 Rachel      40          FALSE

```

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

Answer: This data frame contains multiple types of data, while a matrix can only contain one type.

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.

```

#scores vector from question 5:
scores <- c(80, 90, 70, 40)

#10.
#function that prints "true" if score is 50 or higher or "false" if it is lower than 50
passfail <- function(scores) {
  passfail <- ifelse(scores >= 50, "TRUE", "FALSE")
  print(passfail)
}

#11.
#run function on vector to produce true/false list indicating if each score in the vector was passing
passfail(scores)

```

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

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

Answer: the “ifelse” function worked because it allows you to apply a function to an entire vector, while “if” and “else” do not.