

# Assignment 2: Coding Basics

He Gao

## OVERVIEW

This exercise accompanies the lessons/labs 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 Canvas.
7. Initial here to acknowledge that you did not use AI at all in completing this assignment: \_\_\_\_\_

## Basics, Part 1

1. Use R’s `seq()` function to create a sequence of numbers from 100 to 333, increasing by threes. Assign this sequence a variable name.
2. Compute the *mean* of this sequence, assigning this values its own variable name.
3. Compute the *standard deviation* (`sd()`) of this sequence, assigning this values its own variable name.
4. Display the the mean minus the standard deviation as well as the mean plus the standard deviation.
5. Insert comments in your code to describe what you are doing.

```
# 1. Create a sequence from 100 to 333 increasing by 3
seq_nums <- seq(from = 100, to = 333, by = 3)

# 2. Compute the mean of the sequence
seq_mean <- mean(seq_nums)

# 3. Compute the standard deviation of the sequence
seq_sd <- sd(seq_nums)

# 4. Display mean - sd and mean + sd
seq_mean - seq_sd
```

```
## [1] 147.5184
```

```
seq_mean + seq_sd
```

```
## [1] 283.4816
```

---

## Basics, Part 2

6. Create three vectors, each with four components, consisting of (a) student names, (b) test scores, and (c) whether they are on scholarship or not (TRUE or FALSE).

7. Label each vector with a comment on what type of vector it is.

8. Combine each of the vectors into a data frame. Assign the data frame an informative name.

9. Label the columns of your data frame with informative titles.

```
# (a) Character vector: student names
student_names <- c("Sonia", "Lily", "Mike", "Kevin")

# (b) Numeric vector: test scores
test_scores <- c(88, 76, 59, 93)

# (c) Logical vector: scholarship status (TRUE/FALSE)
on_scholarship <- c(TRUE, FALSE, FALSE, TRUE)

student_df <- data.frame(student_names, test_scores, on_scholarship)
colnames(student_df) <- c("Name", "Score", "Scholarship")
student_df
```

```
##      Name Score Scholarship
## 1 Sonia     88         TRUE
## 2 Lily      76        FALSE
## 3 Mike      59        FALSE
## 4 Kevin     93         TRUE
```

---

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

Answer: A data frame is like a table where each column can have a different type of data (for example, names as text, scores as numbers, and TRUE/FALSE as logical values). However, a matrix can only store one type of data for all elements. Data frames are often used for real-world datasets, while matrices are more common in linear algebra.

---

## Basics, Part 3

11. Create a function with one input. In this function, use `if...else` to evaluate the value of the input: if it is greater than 50, it returns the word “Pass”; otherwise it returns the word “Fail”.
12. Create a second function that does the exact same thing as the previous one but uses `ifelse()` instead of `if...else`.
13. Run both functions using the value 54 as the input
14. Run both functions using the **vector** of student test scores you created as the input. (Only one will work properly...)

*#11. Create a function using if...else*

```
pass_fail_if <- function(x) {  
  if (x > 50) {  
    return("Pass")  
  } else {  
    return("Fail")  
  }  
}
```

*#12. Create a function using ifelse()*

```
pass_fail_ifelse <- function(x) {  
  ifelse(x > 50, "Pass", "Fail")  
}
```

*#13a. Run the first function with the value 54*

```
pass_fail_if(54)
```

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

*#13b. Run the second function with the value 54*

```
pass_fail_ifelse(54)
```

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

*#14a. Run the first function with the vector of test scores (will NOT work)*

```
#pass_fail_if(test_scores)
```

*#14b. Run the second function with the vector of test scores (works)*

```
pass_fail_ifelse(test_scores)
```

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

15. QUESTION: Which option of `if...else` vs. `ifelse` worked? Why? (Hint: search the web for “R vectorization”; it’s ok here if an AI response is presented in the search response.)

Answer: `if...else`: Used for a single condition check (not vectorized), so it cannot directly evaluate an entire column/vector element by element. `ifelse()`: A vectorized conditional function that can apply the condition to each element of a vector at the same time and return an output vector of the same length. The loop-based method checks each element one by one. The vectorized method uses `ifelse()`. It applies the condition to the entire vector at once. In R, vectorized conditionals evaluate conditions over whole vectors, and functions like `ifelse()` apply logic to entire vectors. This simplifies conditional operations.

**NOTE** Before knitting, you'll need to comment out the call to the function in Q14 that does not work. (A document can't knit if the code it contains causes an error!)