

Assignment 2: Coding Basics

Haonan Pei

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. Creating a sequence of numbers from 100 to 333, increasing by 3
seq <- seq(100, 333, 3)
seq
```

```
## [1] 100 103 106 109 112 115 118 121 124 127 130 133 136 139 142 145 148 151 154
## [20] 157 160 163 166 169 172 175 178 181 184 187 190 193 196 199 202 205 208 211
## [39] 214 217 220 223 226 229 232 235 238 241 244 247 250 253 256 259 262 265 268
## [58] 271 274 277 280 283 286 289 292 295 298 301 304 307 310 313 316 319 322 325
## [77] 328 331
```

```
#2. Calculating the mean of this sequence
```

```
seq_mean <- mean(seq)
seq_mean
```

```
## [1] 215.5
```

```
#3. Calculating the standard deviation of this sequence
```

```
seq_sd <- sd(seq)
seq_sd
```

```
## [1] 67.98162
```

```
#4. Calculating and displaying (mean - standard deviation) and (mean + standard deviation)
```

```
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.

```
# "Student names" is a character vector
```

```
student_names <- c("A", "B", "C", "D")
```

```
# "Test scores" is a numeric vector
```

```
test_scores <- c(100, 98, 95, 90)
```

```
# "Whether they are on scholarship or not" is a logical vector
```

```
scholarship <- c(TRUE, FALSE, TRUE, FALSE)
```

```
# Student Information
```

```
student_info <- data.frame(
```

```
  Name = student_names,
```

```
  Score = test_scores,
```

```
  Scholarship = scholarship
```

```
)
```

```
student_info
```

```
##   Name Score Scholarship
```

```
## 1    A   100         TRUE
```

```
## 2    B    98        FALSE
```

```
## 3    C    95         TRUE
```

```
## 4    D    90        FALSE
```

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

Answer: Within a data frame, it can contain different data types. The data frame can be considered as a collection of several different vectors. However, all the elements within a matrix should be the same data type. The matrix is a single vector that arranged into rows and columns.

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
```

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

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

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

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

```
if_function(54)
```

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

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

```
ifelse_function(54)
```

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

```
#14a. Run the first function with the vector of test scores
```

```
#if_function(test_scores)
```

```
#14b. Run the second function with the vector of test scores
```

```
ifelse_function(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: The “ifelse” option worked because this function is “vectorized” so the operation can automatically operate on each element within a vector at the same time. This process checks every value within the vector individually and returns a vector of results.

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!)