Assignment 2: Coding Basics

Humayra Rahman

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.

```r
seq(100, 333)
```

```
##   [1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
##  [19] 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
##  [37] 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153
##  [55] 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
##  [73] 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189
##  [91] 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207
## [109] 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225
## [127] 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243
## [145] 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261
## [163] 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279
## [181] 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297
## [199] 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315
## [217] 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333
```

```r
hundred_sequence <- seq(100, 333,3)
hundred_sequence
```

```
##  [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
```

```r
seq(100, 333, 3) # from, to, by
```

```
##  [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. Compute the *mean* of this sequence, assigning this values its own variable name. mean(hundred_sequence) # Vector type: Numeric
3. Compute the *standard deviation* (`sd()`) of this sequence, assigning this values its own variable name. seq_sd <- sd(hundred_sequence)
4. Display the the mean minus the standard deviation as well as the mean plus the standard deviation. mean(hundred_sequence)-seq_sd mean(hundred_sequence)+seq_sd
5. Insert comments in your code to describe what you are doing.

```r
#1. hundred_sequence <- seq(from = 100, to = 333, by = 3)

#2. seq_mean <- mean(hundred_sequence)

#3. seq_sd <- sd(hundred_sequence)

#4.seq_mean - seq_sd; seq_mean + seq_sd
```

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). student_names <- c("Anna", "Ben", "Cary", "Don") # Vector type: character test_scores <- c(75, 88, 95, 80) # Vector type: numeric scholarship <- c(FALSE, TRUE, TRUE, FALSE) # Vector type: logic
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. student_data <- data.frame(student_names,test_scores, scholarship)
9. Label the columns of your data frame with informative titles. student_data <- data.frame(Student_Name = student_names, Test_Score = test_scores, Scholarship_Status = scholarship)

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

    Answer: A data frame can hold different types of data in different columns, like names, numbers, and TRUE/FALSE values all together. A matrix can only store one type of data at a time. It can be numeric, character, or logical, but not a mix of all three at once.

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

pass_fail(83) pass_fail(35)

12. Create a second function that does the exact same thing as the previous one but uses `ifelse()` instead if `if...else`. pass_fail_ifelse <- function(x) { ifelse(x > 50, "Pass", "Fail") }

pass_fail_ifelse(50)

13. Run both functions using the value 54 as the input pass_fail(54);pass_fail_ifelse(54)
14. Run both functions using the **vector** of student test scores you created as the input. (Only one will work properly...)

pass_fail_ifelse(test_scores)

```r
#11. Create a function using if...else
pass_fail <- function(x) {
  if (x > 60) {
    return("Pass")
  } else {
    return("Fail")
  }
}

#12. Create a function using ifelse()
pass_fail_ifelse <- function(x) {
  ifelse(x > 60, "Pass", "Fail")
}
#13a. Run the first function with the value 54
pass_fail <- function(x) {
  if (x > 54) {
    return("Pass")
  } else {
    return("Fail")
  }
}

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

```
pass_fail_ifelse <- function(x)
  ifelse(x > 54, "Pass", "Fail")

#14a. Run the first function with the vector of test scores
  pass_fail <- function(test_scores) {
  if (x > 54) {
    return("Pass")
  } else {
    return("Fail")
  }
}

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

pass_fail_ifelse <- function(test_scores) { ifelse(x > 54, “Pass”, “Fail”)

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: ifelse() worked. It worked because ifelse() is vectorized, meaning it checks each value in the vector one by one and returns a result for each value. The if … else statement only works with a single value, not a vector, so it does not work properly when given multiple values at once.

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!)