Assignment 1

Yesdi Christian Calvin

2023-09-05

Git and GitHub

1) Provide the link to the GitHub repo that you used to practice git from Week 1. It should have:

- Your name on the README file.
- At least one commit with your name, with a description of what you did in that commit.

=> This is the link

Reading Data

Download both the Angell.dta (Stata data format) dataset and the Angell.txt dataset from this website: https://stats.idre.ucla.edu/stata/examples/ara/applied-regression-analysis-by-fox-data-files/

2) Read in the .dta version and store in an object called angell_stata.

```
library(haven)
angell_stata<-read_dta("E:/Umich/727/Class 2 Assignment 1/angell.dta")
angell_stata
```

```
## # A tibble: 43 x 5
##    city       morint ethhet geomob region
##    <chr>       <dbl>  <dbl>  <dbl> <chr>
##  1 Rochester    19     20.6   15   E
##  2 Syracuse     17     15.6   20.2 E
##  3 Worcester    16.4   22.1   13.6 E
##  4 Erie         16.2   14     14.8 E
##  5 Milwaukee    15.8   17.4   17.6 MW
##  6 Bridgeport   15.3   27.9   17.5 E
##  7 Buffalo      15.2   22.3   14.7 E
##  8 Dayton       14.3   23.7   23.8 MW
##  9 Reading      14.2   10.6   19.4 E
## 10 Des_Moines   14.1   12.7   31.9 MW
## # i 33 more rows
```

3) Read in the .txt version and store it in an object called angell_txt.

```
angell_txt<-read.table("E:/Umich/727/Class 2 Assignment 1/angell.txt")
angell_txt
```

```
##                 V1   V2   V3   V4 V5
## 1        Rochester 19.0 20.6 15.0  E
## 2         Syracuse 17.0 15.6 20.2  E
## 3        Worcester 16.4 22.1 13.6  E
## 4             Erie 16.2 14.0 14.8  E
## 5        Milwaukee 15.8 17.4 17.6 MW
## 6       Bridgeport 15.3 27.9 17.5  E
## 7          Buffalo 15.2 22.3 14.7  E
## 8           Dayton 14.3 23.7 23.8 MW
## 9          Reading 14.2 10.6 19.4  E
## 10      Des_Moines 14.1 12.7 31.9 MW
## 11       Cleveland 14.0 39.7 18.6 MW
## 12          Denver 13.9 13.0 34.5  W
## 13          Peoria 13.8 10.7 35.1 MW
## 14         Wichita 13.6 11.9 42.7 MW
## 15         Trenton 13.0 32.5 15.8  E
## 16    Grand_Rapids 12.8 15.7 24.2 MW
## 17          Toledo 12.7 19.2 21.6 MW
## 18       San_Diego 12.5 15.9 49.8  W
## 19       Baltimore 12.0 45.8 12.1  E
## 20      South_Bend 11.8 17.9 27.4 MW
## 21           Akron 11.3 20.4 22.1 MW
## 22         Detroit 11.1 38.3 19.5 MW
## 23          Tacoma 10.9 17.8 31.2  W
## 24           Flint  9.8 19.3 32.2 MW
## 25         Spokane  9.6 12.3 38.9  W
## 26         Seattle  9.0 23.9 34.2  W
## 27    Indianapolis  8.8 29.2 23.1 MW
## 28        Columbus  8.0 27.4 25.0 MW
## 29 Portland_Oregon  7.2 16.4 35.8  W
## 30        Richmond 10.4 65.3 24.9  S
## 31         Houston 10.2 49.0 36.1  S
## 32      Fort_Worth 10.2 30.5 36.8  S
## 33   Oklahoma_City  9.7 20.7 47.2  S
## 34     Chattanooga  9.3 57.7 27.2  S
## 35       Nashville  8.6 57.4 25.4  S
## 36      Birmingham  8.2 83.1 25.9  S
## 37          Dallas  8.0 36.8 37.8  S
## 38      Louisville  7.7 31.5 19.4  S
## 39    Jacksonville  6.0 73.7 27.7  S
## 40         Memphis  5.4 84.5 26.7  S
## 41           Tulsa  5.3 23.8 44.9  S
## 42           Miami  5.1 50.2 41.8  S
## 43         Atlanta  4.2 70.6 32.6  S
```

4) What are the differences between angell_stata and angell_txt? Are there differences in the classes of the individual columns?

```r
#check the class of angell_stata
class(angell_stata)
```

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

```r
#check the class of angell_txt
class(angell_txt)
```

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

Yes, there are differences between the labels or names of the individual columns. In angell_stat, each column has a specific label/name that represents the variable's name. At the same time, angell_txt does not contain a label for every column. So, when RStudio reads the file, all the labels are automatically generated with default names, which are V1, V2, V3, V4, and V5. In addition, they have different types of classes.

5) Make any updates necessary so that angell_txt is the same as angell_stata.

```r
library(tibble)

#change the angell_txt from data frame to tibble
angell_txt <- as_tibble(angell_txt)
#check the data format of angell_txt
class(angell_txt)
```

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

Both have the same class now.

```r
## set column names using the colnames() function.

colnames(angell_txt) <- c("city", "morint", "ethhet", "geomob", "region")

print(angell_txt)
```

```
## # A tibble: 43 x 5
##    city       morint ethhet geomob region
##    <chr>       <dbl>  <dbl>  <dbl> <chr>
##  1 Rochester    19     20.6   15   E
##  2 Syracuse     17     15.6   20.2 E
##  3 Worcester    16.4   22.1   13.6 E
##  4 Erie         16.2   14     14.8 E
##  5 Milwaukee    15.8   17.4   17.6 MW
##  6 Bridgeport   15.3   27.9   17.5 E
##  7 Buffalo      15.2   22.3   14.7 E
##  8 Dayton       14.3   23.7   23.8 MW
##  9 Reading      14.2   10.6   19.4 E
## 10 Des_Moines   14.1   12.7   31.9 MW
## # i 33 more rows
```

Both have the same labels for each column now.

6) Describe the Ethnic Heterogeneity variable. Use descriptive statistics such as mean, median, standard deviation, etc. How does it differ by region?

```r
# Use summary() to provide a quick summary of several statistics
summary(angell_txt$ethhet)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.60   16.90   23.70   31.37   39.00   84.50
```

```r
#Use var() to calculate the sample variance of a numeric vector.
var(angell_txt$ethhet)
```

```
## [1] 416.6287
```

```r
#Use sd() to calculate the standard deviation of a numeric vector.
sd(angell_txt$ethhet)
```

```
## [1] 20.41149
```

```r
#How does it differ by region?
#the first thing we can do is activating the dplyr package, so we can use a syntax named group_by

library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
#after dplyr activated, we can use verb "group_by" to display some descriptive statistics by different

angell_txt%>%
group_by(region)%>%
summarise(mn = mean(ethhet),
          md= median(ethhet),
          var_ethhet = var(ethhet),
          se_ethhet = sqrt(var_ethhet))
```

```
## # A tibble: 4 x 5
##   region    mn    md var_ethhet se_ethhet
##   <chr>  <dbl> <dbl>      <dbl>     <dbl>
## 1 E       23.5  22.1      116.      10.8
## 2 MW      21.7  19.2       82.5      9.08
## 3 S       52.5  53.8      460.      21.4
## 4 W       16.6  16.2       17.3      4.16
```

Comparison of the statistics among the groups:

Compared to other regions, the Southeast has a moderately higher ethnic heterogeneity. The mean percentage of ethnic heterogeneity at the national level (31.37) is even lower than the percentage of ethnic heterogeneity in the Southeast area (52.48). Compared to other regions, the West has the lowest rate of variability. Moreover, the mean and median difference is often tiny across all regions. It denotes the absence of extreme values of each region's data's ethnic heterogeneity. The mean and median being similar can sometimes be an indication of normal distribution. However, it is not a definitive test for normality. In addition, the standard deviation is entirely consistent across all regions.

Describing Data

R comes also with many built-in datasets. The "MASS" package, for example, comes with the "Boston" dataset.

7) Install the "MASS" package, load the package. Then, load the Boston dataset.

```
#load the "MASS" package
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
#load Boston dataset
data(Boston)
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

8) What is the type of the Boston object?

```
#determine the data type of Boston object using "typeof" function
typeof(Boston)
```

```
## [1] "list"
```

It is shown that the data type of Boston object is list.

```
#access Boston attributes associated with the object
attributes(Boston)
```

```
## $names
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
##
## $class
## [1] "data.frame"
##
## $row.names
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
## [109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
## [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
## [145] 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
## [163] 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
## [181] 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
## [199] 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
## [217] 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
## [235] 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
## [253] 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
## [271] 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
## [289] 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
## [307] 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
## [325] 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
## [343] 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
## [361] 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
## [379] 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
## [397] 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414
## [415] 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
## [433] 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450
## [451] 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468
## [469] 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486
## [487] 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504
## [505] 505 506
```

9) What is the class of the Boston object?

```
#determine the class of Boston object using "class" function
class(Boston)
```

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

```
class(1:10)
```

```
## [1] "integer"
```

It is shown that the class of Boston object is data frame containing integer values

10) How many of the suburbs in the Boston data set bound the Charles river?

```
# Count the number of units with a value of 1 (if tract bounds river) in the dummy variable
count_units <- sum(Boston$chas == 1)

# Display the count
print(count_units)
```

```
## [1] 35
```

The number of suburbs that bound the charles river is 35.

11) Do any of the suburbs of Boston appear to have particularly high crime rates? Tax rates? Pupil-teacher ratios? Comment on the range of each variable.

To show the range of each variable, we can use "summary" function to display the Maximum and Minimum value.

```
#check the range of crime
summary(Boston$crim)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##   0.00632   0.08204   0.25651   3.61352   3.67708 88.97620
```

The Mean and the 3rd Quartile show that the crime rates are below 4, meaning that overall, the suburbs in Boston have pretty low crime rates, with the lowest crime rate reaching 0.00632. However, the highest crime rate is 88.9720. It indicates that there are suburbs that have an extremely high crime rate. In brief, there must be suburbs with high crime rates, but the suburbs with low crime rates are more dominant in Boston.

```
#check the range of tax
summary(Boston$tax)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   187.0   279.0   330.0   408.2   666.0   711.0
```

The range for tax rates is 524, calculated from a maximum value of 711 and a minimum of 187. If we compare these two values, we can conclude that there are suburbs' that are significantly higher than others regarding tax rates. On the other hand, judging from the quartiles, mean, and median, the data are dispersed over a broad interval.

```
#check the range of pupil-teacher ratio
summary(Boston$ptratio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.60   17.40   19.05   18.46   20.20   22.00
```

The range for pupil-teacher ratio is 9.4, calculated from a maximum value of 22 and a minimum of 12.6. Comparing these two values, we can conclude that there are suburbs' that are higher than others regarding pupil-teacher rates. The gap between the mean and the max also says that. However, the range is relatively close, indicating that Boston probably has no suburbs with extremely high pupil-teacher ratios.

12) Describe the distribution of pupil-teacher ratio among the towns in this data set that have a per capita crime rate larger than 1. How does it differ from towns that have a per capita crime rate smaller than 1?

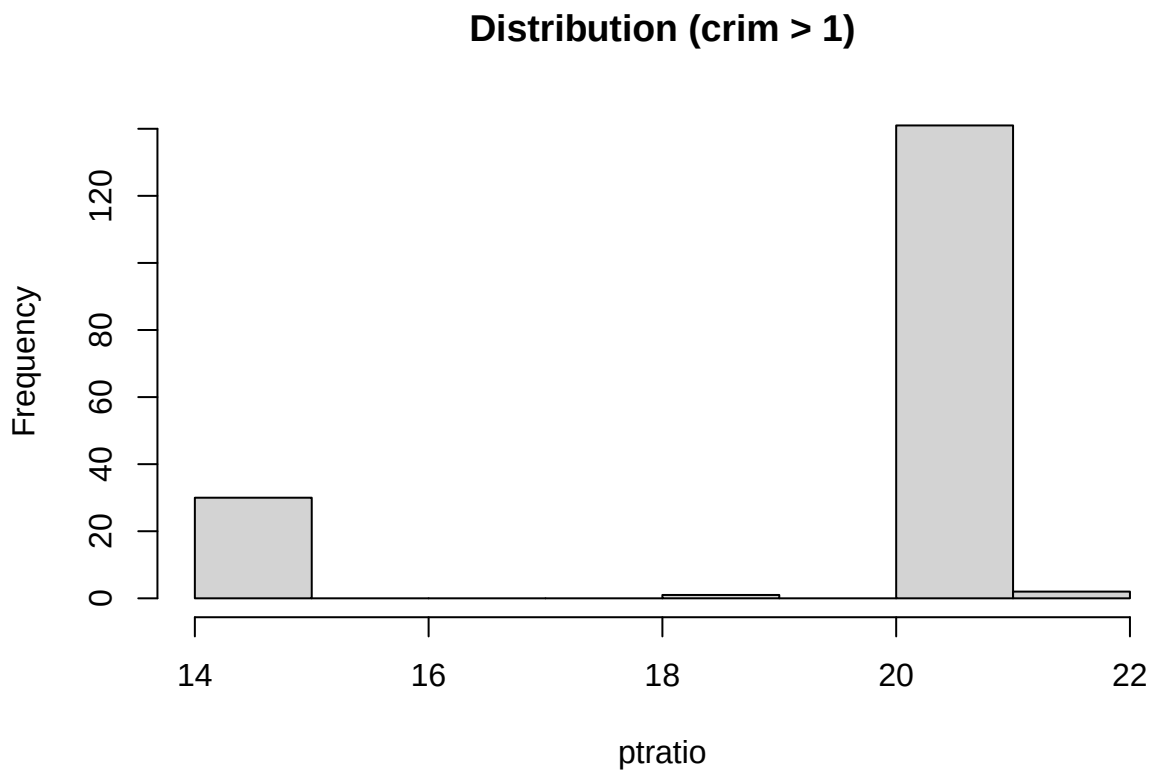
```r
# First of all, we have to form two different subsets based on the dummy variable
# Subset for values > 1
subset_large <- Boston[Boston$crim > 1, ]

# Subset for values < 1
subset_small <- Boston[Boston$crim < 1, ]
```

```r
# Descriptive statistics for values > 1
summary(subset_large$ptratio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.70   20.20   20.20   19.29   20.20   21.20
```

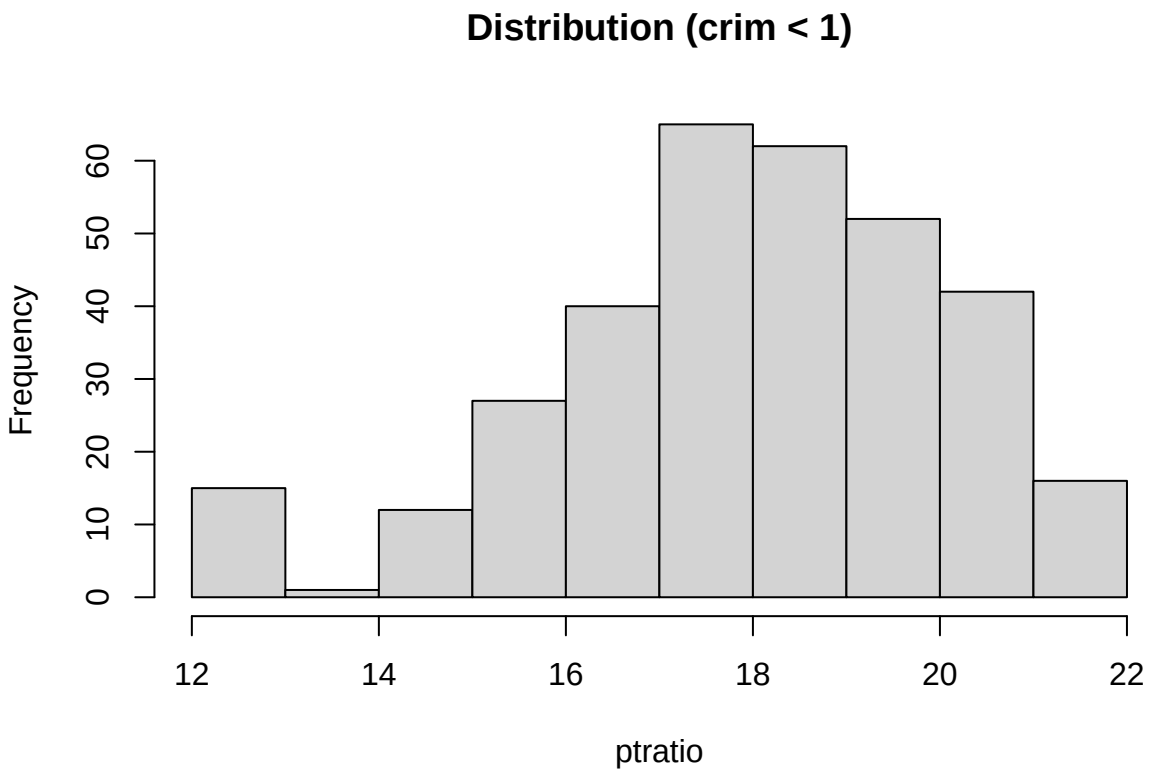
```r
hist(subset_large$ptratio, main="Distribution (crim > 1)", xlab="ptratio")
```




```r
# Descriptive statistics for values < 1
summary(subset_small$ptratio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.60   16.80   18.30   18.02   19.20   22.00
```

```r
hist(subset_small$ptratio, main="Distribution (crim < 1)", xlab="ptratio")
```



Based on the histogram, it can be seen that the pupil-teacher ratios in suburbs with a crime rate above 1 tend to be centered at a certain point, or in other words, many of these suburbs have similar pupil-teacher ratios. It is supported by the figures from the summary of descriptive statistics that the gap between the quartiles, mean, median and max values is very small. In other words, the values are relatively similar.

On the other hand, pupil-teacher ratio data in suburbs with a crime rate below 1 tends to spread over wider intervals.

Writing Functions

13) Write a function that calculates 95% confidence intervals for a point estimate. The function should be called my_CI. When called with my_CI(2, 0.2), the function should print out "The 95% CI upper bound of point estimate 2 with standard error 0.2 is 2.392. The lower bound is 1.608."

Note: The function should take a point estimate and its standard error as arguments. You may use the formula for 95% CI: point estimate +/- 1.96*standard error.

Hint: Pasting text in R can be done with: paste() and paste0()

```
my_CI <- function(point_etimate, se){

  # Calculate the lower and upper bounds of the 95% CI
    lower_bound <- point_etimate - 1.96 * se
    upper_bound <- point_etimate + 1.96 * se

  # Calling the lower and upper bounds as a text using "paste" function
```

```
    text <- paste("The 95% CI upper bound of point estimate 2 with standard error 0.2 is", upper_bound,
}

#CI of point estimate 2 and standard error 0.2
CI <- my_CI(2,0.2)

#print the result
CI
```

```
## [1] "The 95% CI upper bound of point estimate 2 with standard error 0.2 is 2.392 . The lower bound is
```

14) Create a new function called my_CI2 that does that same thing as the my_CI function but outputs a vector of length 2 with the lower and upper bound of the confidence interval instead of printing out the text. Use this to find the 95% confidence interval for a point estimate of 0 and standard error 0.4.

```
my_CI2 <- function(point_etimate, se){

  # Calculate the lower and upper bounds of the 95% CI
    lower_bound <- point_etimate - 1.96 * se
    upper_bound <- point_etimate + 1.96 * se

  # Return the lower and upper bounds as a vector
    return(c(lower_bound, upper_bound))
}

#CI with standard error 0.4
CI2 <- my_CI2(0,0.4)

#print the result
CI2
```

```
## [1] -0.784  0.784
```

15) Update the my_CI2 function to take any confidence level instead of only 95%. Call the new function my_CI3. You should add an argument to your function for confidence level.

Hint: Use the qnorm function to find the appropriate z-value. For example, for a 95% confidence interval, using qnorm(0.975) gives approximately 1.96.

```
my_CI3 <- function(point_estimate, se, confidence_level) {

  # Calculate the z-value for the desired confidence level
    z <- qnorm(1 - (1 - confidence_level) / 2)

  # Calculate the lower and upper bounds of the CI
    lower_bound <- point_estimate - z * se
    upper_bound <- point_estimate + z * se

  # Return the lower and upper bounds as a vector
    return(c(lower_bound, upper_bound))
}
```

```r
# Example usage with a 99% confidence level:
CI3 <- my_CI3(0,0.4,0.99)

#print the result
print(CI3)
```

```
## [1] -1.030332  1.030332
```

16) Without hardcoding any numbers in the code, find a 99% confidence interval for Ethnic Heterogeneity in the Angell dataset. Find the standard error by dividing the standard deviation by the square root of the sample size.

Hardcoding numbers in the code means directly specifying numerical values within the code itself, without using variables or expressions to represent those values dynamically. Therefore, in this case, I use variables or expressions to represent values, which makes my code more flexible.

```r
# Compute sample mean and standard deviation
mean_val <- mean(angell_stata$ethhet)
std_dev <- sd(angell_stata$ethhet)

#Calculate the sample size (n) by counting the number of observations
n <- length(angell_stata$ethhet)

#Calculate the standard error (SE) by dividing the standard deviation by the square root of the sample
SE <- std_dev / sqrt(n)

#Find the critical value corresponding to a 99% confidence interval. Use the qnorm() function to find t
alpha <- 0.01  # 1 - Confidence level
z <- qnorm(1 - alpha/2)  # For a two-tailed test

#Calculate the margin of error (MOE) by multiplying the critical value by the SE
MOE <- z * SE

#Calculate the 99% CI by adding and subtracting the MOE from the sample mean
lower_bound <- mean_val - MOE
upper_bound <- mean_val + MOE

#print the result
cat(lower_bound,upper_bound)
```

```
## 23.35425 39.38993
```

17) Write a function that you can apply to the Angell dataset to get 95% confidence intervals. The function should take one argument: a vector. Use if-else statements to output NA and avoid error messages if the column in the data frame is not numeric or logical.

```r
#Create a function to get 95% CI
angell_95_CI <- function(vector) {

  # Check if the input vector is numeric or logical using if-else statement
  if (is.numeric(vector)){
```

```
      # Calculate the mean and standard error
      mean_val <- mean(vector)
      se <- sd(vector) / sqrt(length(vector))

      # Calculate the critical value for a 95% confidence interval
      z <- qnorm(0.975)

      # Calculate the lower and upper bounds of the CI
      lower_bound <- mean_val - z * se
      upper_bound <- mean_val + z * se

      # Return the confidence interval as a vector
      return(c(lower_bound, upper_bound))
      }
  else {
  #return NA if the data is non-numeric
  return(NA)
  }
}

#create CI vectors of the angell_txt and return as list using the "lapply" function
return_list <- lapply(angell_txt, angell_95_CI)
return_list
```

```
## $city
## [1] NA
##
## $morint
## [1] 10.13242 12.26758
##
## $ethhet
## [1] 25.27127 37.47292
##
## $geomob
## [1] 24.67187 30.52347
##
## $region
## [1] NA
```