

Problem Set 1

Jacob Hood

February 27th, 2021

Good job! Your code is very clean and well organized.

For part 2, please add in-line comments to explain your steps in the future..

For part 3, the se and CI for one sample is incorrect because the length() function doesn't give you the number of observations for a dataframe. Use nrow() instead. The questions asking you to estimate se using the sampling distribution should use the sd of the sampling distribution.

Overall great work!

92/100

```
# Question 1
```

```
# 1.1: Formula for population mean
```

```
pop_mean <- function(x) {  
  out_vector <- sum(x)/length(x) # divide sum by total number of observations  
  return(out_vector)  
}
```

```
# 1.2: Formula to calculate variance
```

```
pop_var <- function(x) {  
  var = sum((x - pop_mean(x))^2)/(length(x)) #(sum of each number - the mean)^2/sample size  
  return(var)  
}
```

```
# 1.3
```

```
gapminder <- read.csv("data/gapminder.csv")
```

```
pop_mean(gapminder$lifeExp)
```

```
## [1] 59.47444
```

```
pop_var(gapminder$lifeExp)
```

```
## [1] 166.7537
```

```
mean(gapminder$lifeExp)
```

```
## [1] 59.47444
```

```
var(gapminder$lifeExp)
```

```
## [1] 166.8517
```

```
# The mean value for both functions is 59.474. The variance for the
# function I wrote is 166.753. The variance for the R function is
# 166.852. They are different because my function calculates
# population variance, while R's function calculates sample variance.
```

```
# Question #2
```

```
# Import data
```

```
parent_inc <- read.csv("data/parent_inc.csv")
```

```
# Tidy data
```

```
parent_inc %>% rename(father_income = fincome, mother_income = mincome) %>%
  pivot_longer(cols = c(father_name, mother_name, father_income, mother_income),
    names_to = c("type", ".value"), names_sep = "_")
```

Please comment your steps next time.

```
## # A tibble: 6 x 4
##   famid type   name  income
##   <int> <chr>  <chr>   <int>
## 1     1 father Arthur  42000
## 2     1 mother Jess   45000
## 3     2 father Harry  35000
## 4     2 mother Pam    24000
## 5     3 father Matt   78000
## 6     3 mother Mary   55000
```

Add set.seed() here

```
# Question #3 3.1: Create a population data frame
```

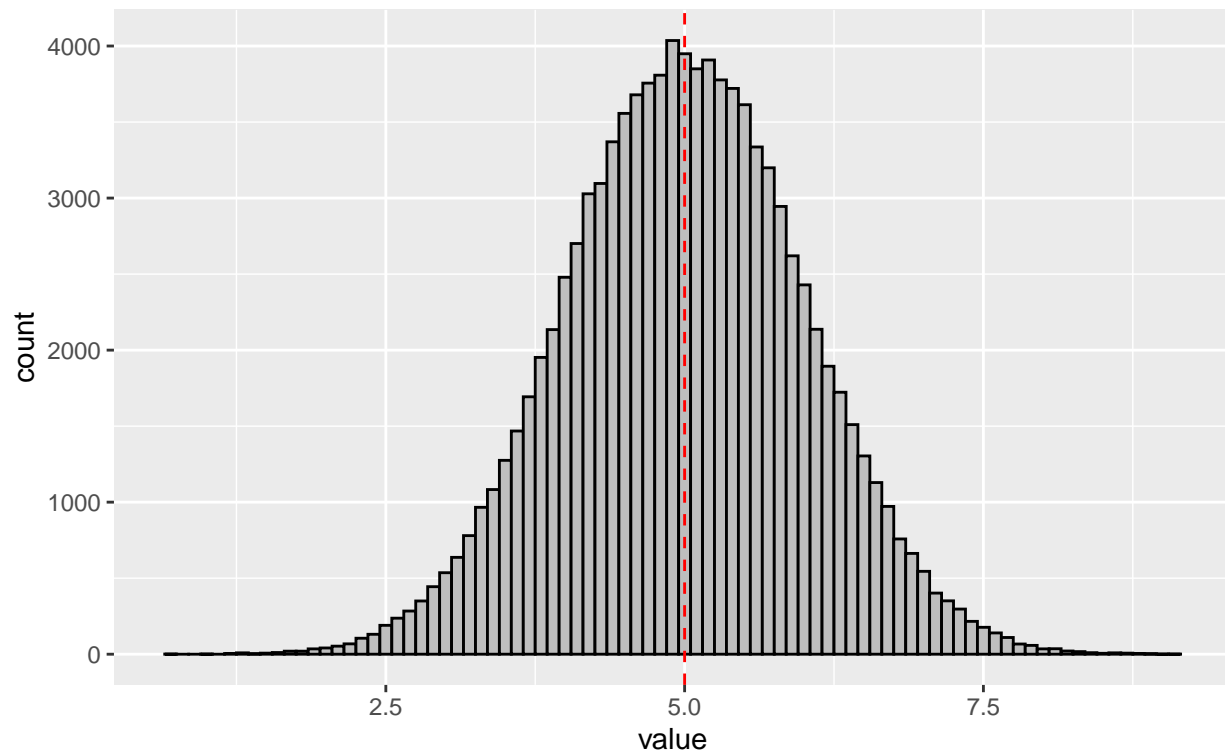
```
value <- rnorm(n = 1e+05, mean = 5, sd = 1) %>% as_tibble()
```

```
# 3.2: Create a histogram
```

```
value %>% ggplot(aes(value)) + geom_histogram(binwidth = 0.1, color = "black",
  fill = "grey") + geom_vline(aes(xintercept = mean(value)), color = "red",
  linetype = "dashed") + labs(title = "Histogram of Simulated Population with Normal Distribution",
  subtitle = "N = 100000, SD = 1")
```

Histogram of Simulated Population with Normal Distribution

N = 100000, SD = 1



3.3: Draw random sample n=50

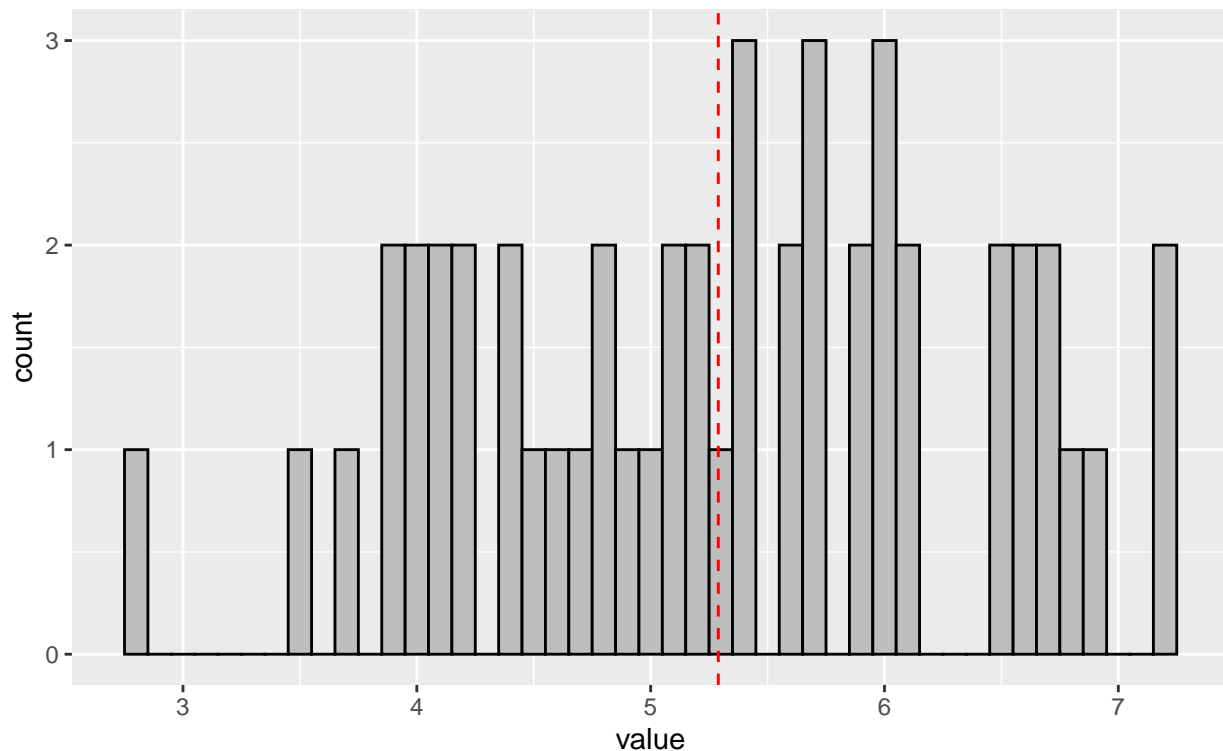
```
set.seed(110001)
sample <- value %>% sample_n(size = 50, replace = FALSE)
```

3.4: Plot a histogram of the sample

```
sample %>% ggplot(aes(value)) + geom_histogram(binwidth = 0.1, color = "black",
  fill = "grey") + geom_vline(aes_string(xintercept = paste0("mean(",
  sample, ")")), color = "red", linetype = "dashed") + labs(title = "Histogram of Random Sample",
  subtitle = "N = 50")
```

Histogram of Random Sample

N = 50



3.5:

Calculate point estimate for population mean

```
point_est <- sum(sample)/count(sample)
print(point_est)
```

```
##          n
## 1 5.289375
```

Calculate standard error of point estimate

```
standard_error <- sapply(sample, sd)/sqrt(length(sample))
print(standard_error)
```

```
##      value
## 1 1.053264
```

the standard error value is off, length(sample) =1 because sample is a dataframe. Use nrow(sample) instead.

Calculate 95% confidence interval of point estimate

*# The formula is: population mean +/- t-critical value * standard error*

Calculate critical value

```
observed_t <- point_est/standard_error
print(observed_t)
```

```
##          n
## 1 5.021889
```

```
crit_val <- qt(p = 0.5 * 0.05, df = 49)
print(crit_val)
```

```
## [1] -2.009575
```

```
# Caclulate confidence interval
```

```
upper_bound <- as.numeric(point_est) - (as.numeric(crit_val) * as.numeric(standard_error))
lower_bound <- as.numeric(point_est) + (as.numeric(crit_val) * as.numeric(standard_error))
CI <- c(lower_bound, upper_bound)
print(CI)
```

```
## [1] 3.172762 7.405989
```

This is off b/c se is incorrect.

```
# 3.6 Simulate the sampling distribution of the sample mean (n = 50)
# using 1,000 draws
```

```
mean_container <- vector(mode = "numeric", length = 1000)

set.seed(110001)
for (i in 1:1000) {
  sample_two <- value %>% sample_n(size = 50, replace = FALSE)
  mean_container[i] <- mean(sample_two$value)
}

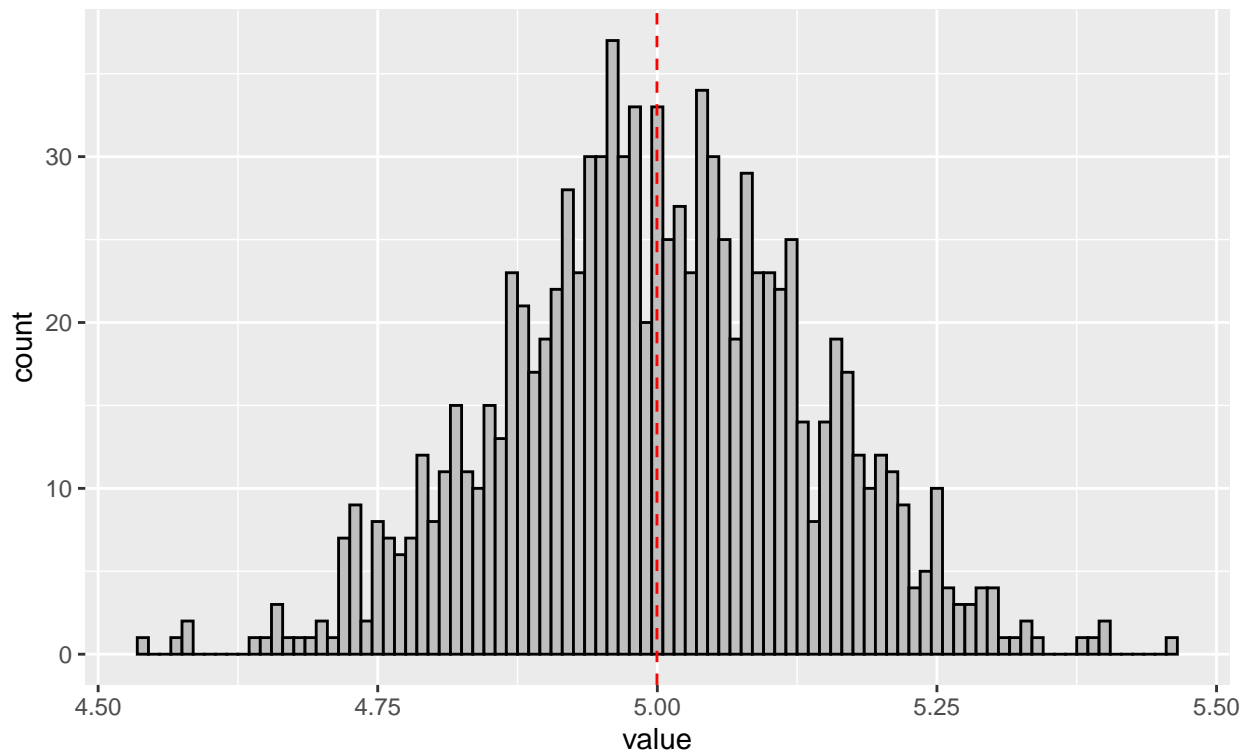
head(mean_container, n = 10)
```

```
## [1] 5.289375 4.889904 5.094229 5.059246 4.829965 5.017235 4.917106 5.194634
## [9] 5.021282 4.888220
```

```
# 3.7 Create a histogram of the sampling distribution of the sample
# mean
```

```
mean_container %>% as_tibble %>% ggplot(aes(value)) + geom_histogram(binwidth = 0.01,
  color = "black", fill = "grey") + geom_vline(aes_string(xintercept = mean(mean_container)),
  color = "red", linetype = "dashed") + labs(title = "Histogram of Sampling Distribution of Sample Mean",
  subtitle = "N = 50")
```

Histogram of Sampling Distribution of Sample Mean Using 1,000 Draws
N = 50



3.8 Calculate point estimate for population mean

```
point_est_two <- sum(mean_container)/length(mean_container)
print(point_est_two)
```

```
## [1] 4.999681
```

Calculate standard error of point estimate

Sample standard deviation/square root of sample size

```
standard_error_two <- sd(mean_container)/sqrt(length(mean_container))
print(standard_error_two)
```

```
## [1] 0.004349864
```

Here the question ask you to estimate the se for the “sample” mean. The standard error of the sample mean is the sd of the sampling distribution.

Calculate 95% confidence interval of point estimate

*# The formula is: population mean +/- t-critical value * standard error*

Calculate critical value

```
observed_t_two <- point_est_two/standard_error_two
print(observed_t_two)
```

```
## [1] 1149.388
```

```
crit_val_two <- qt(p = 0.5 * 0.05, df = 49)
print(crit_val_two)
```

```
## [1] -2.009575
```

```
# Caclulate confidence interval
```

```
upper_bound_two <- as.numeric(point_est_two) - (as.numeric(crit_val_two) *
  as.numeric(standard_error_two))
lower_bound_two <- as.numeric(point_est_two) + (as.numeric(crit_val_two) *
  as.numeric(standard_error_two))
CI_two <- c(lower_bound_two, upper_bound_two)
print(CI_two)
```

```
## [1] 4.990939 5.008422
```

```
# 3.9 Repeat questions 3 to 8 increasing sample size to n=1000
```

```
# a) Draw a random sample n=1,000
```

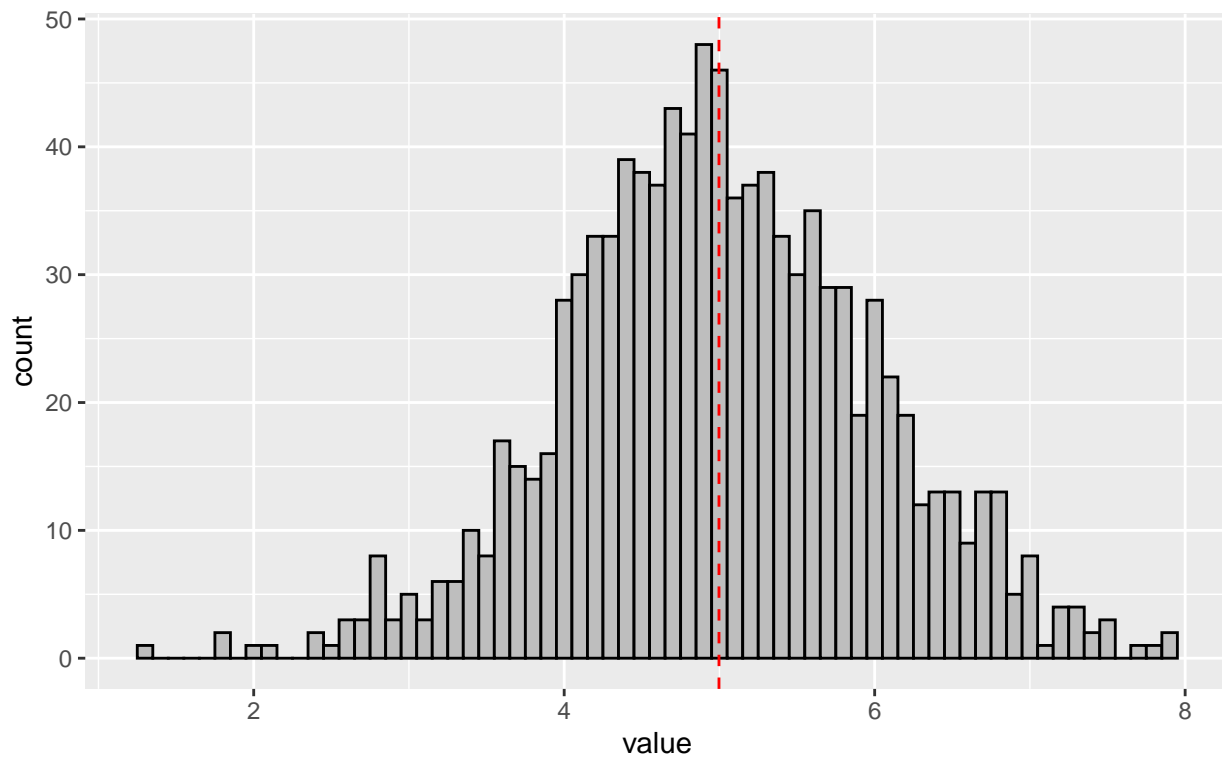
```
set.seed(110001)
final_sample <- value %>% sample_n(size = 1000, replace = FALSE) %>% as_tibble()
```

```
# b) Plot a histogram of the sample with appropriate title and labels
```

```
final_sample %>% ggplot(aes(value)) + geom_histogram(binwidth = 0.1, color = "black",
  fill = "grey") + geom_vline(aes_string(xintercept = paste0("mean(",
  final_sample, ")")), color = "red", linetype = "dashed") + labs(title = "Histogram of Random Sample",
  subtitle = "N = 1000")
```

Histogram of Random Sample

N = 1000



```
# c) Calculate point estimate
```

```
point_est_three <- sum(final_sample)/count(final_sample)
print(point_est_three)
```

```
##           n
## 1 4.997173
```

```
# Calculate standard error of point estimate
```

```
standard_error_three <- sapply(final_sample, sd)/sqrt(length(final_sample))
print(standard_error_three)
```

```
##      value
## 0.9931817
```

same problem as previous question. use `nrow(final_sample)` to get the sample size `n`

```
# Calculate 95% confidence interval of point estimate The formula is:
# population mean +/- t-critical value * standard error
```

```
# Calculate critical value
```

```
observed_t_three <- point_est_three/standard_error_three
print(observed_t_three)
```



```
##          n
## 1 5.031479
```

```
crit_val_three <- qt(p = 0.5 * 0.05, df = 999)
print(crit_val_three)
```

```
## [1] -1.962341
```

```
# Caclulate confidence interval
```

```
upper_bound_three <- as.numeric(point_est_three) - (as.numeric(crit_val_three) *
  as.numeric(standard_error_three))
lower_bound_three <- as.numeric(point_est_three) + (as.numeric(crit_val_three) *
  as.numeric(standard_error_three))
CI_three <- c(lower_bound_three, upper_bound_three)
print(CI_three)
```

```
## [1] 3.048211 6.946135
```

```
# d)
```

```
final_mean_container <- vector(mode = "numeric", length = 1000)

set.seed(110001)
for (i in 1:1000) {
  sample_three <- value %>% sample_n(size = 1000, replace = FALSE)
  final_mean_container[i] <- mean(sample_three$value)
}

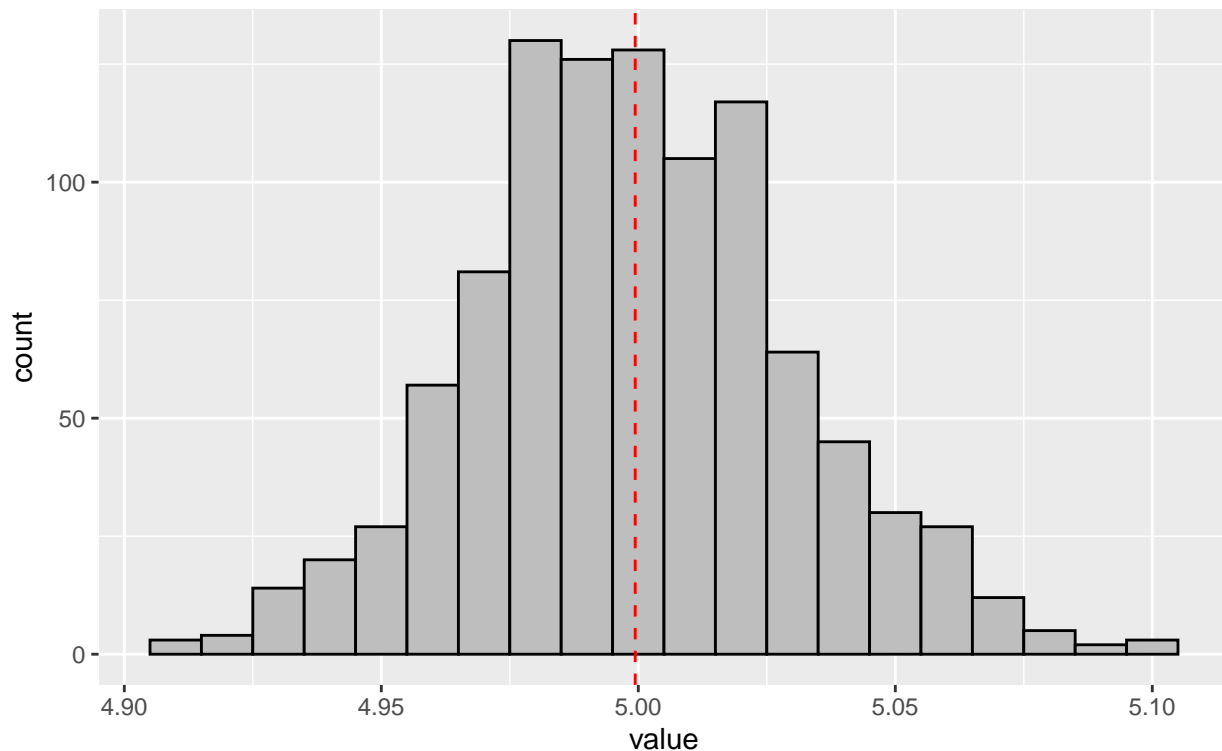
head(final_mean_container, n = 10)
```

```
## [1] 4.997173 4.984034 4.958647 5.031017 4.990072 4.976581 4.993773 5.004962
## [9] 5.005620 4.996527
```

```
# e) Create a histogram of the sampling distribution of the sample mean
# you simulated
```

```
final_mean_container %>% as_tibble %>% ggplot(aes(value)) + geom_histogram(binwidth = 0.01,
  color = "black", fill = "grey") + geom_vline(aes_string(xintercept = mean(final_mean_container)),
  color = "red", linetype = "dashed") + labs(title = "Histogram of Sampling Distribution of Sample Mean",
  subtitle = "N = 1000")
```

Histogram of Sampling Distribution of Sample Mean Using 1,000 Draws
N = 1000



f.1) Calculate point estimate for population mean

```
point_est_four <- sum(final_mean_container)/length(final_mean_container)
print(point_est_four)
```

```
## [1] 4.999397
```

f.2) Calculate standard error of point estimate

Sample standard deviation/square root of sample size

```
standard_error_four <- sd(final_mean_container)/sqrt(length(final_mean_container))
print(standard_error_four)
```

```
## [1] 0.0009872101
```

Same issue as the n = 50 question.

f.3) Calculate 95% confidence interval of point estimate

*# The formula is: population mean +/- t-critical value * standard error*

Calculate critical value

```
observed_t_four <- point_est_four/standard_error_four
print(observed_t_four)
```

```
## [1] 5064.167
```

```
crit_val_four <- qt(p = 0.5 * 0.05, df = 999)
print(crit_val_four)
```

```
## [1] -1.962341
```

```
# Caclulate confidence interval
```

```
upper_bound_four <- as.numeric(point_est_four) - (as.numeric(crit_val_four) *
  as.numeric(standard_error_four))
lower_bound_four <- as.numeric(point_est_four) + (as.numeric(crit_val_four) *
  as.numeric(standard_error_four))
CI_four <- c(lower_bound_four, upper_bound_four)
print(CI_four)
```

```
## [1] 4.997460 5.001334
```

```
# The concept being demonstrated here is the (Weak) Law of Large
# Numbers. As the sample size increases, and the number of samples
# draws increases, the sample mean more nearly equals the population
# mean. This is evidenced by the smaller standard errors and smaller
# confidence intervals, showing that the estimator became more precise.
```