Plotting Exercises Solutions

Christopher Wong

Exercise 1

Central Limit Theorem graphically

We will prove that the Central Limit Theorem (CLT) holds well for $n \geq 30$ graphically.

- a.) Take samples of size **m** from a distribution of your choice (you choose the parameters as well).

```
# In case anyone wants to follow along.
set.seed(123456)
```

```
# I will use the Poisson distribution.
vector.length <- 10
parameter <- 20
poisson.sample <- rpois(vector.length, parameter)
poisson.sample
```

```
##  [1] 23 18 18 20 30 23 25 31 25 18
```

- b.) Find the mean of your random sample.

```
sample.mean <- mean(poisson.sample)
sample.mean
```

```
## [1] 23.1
```

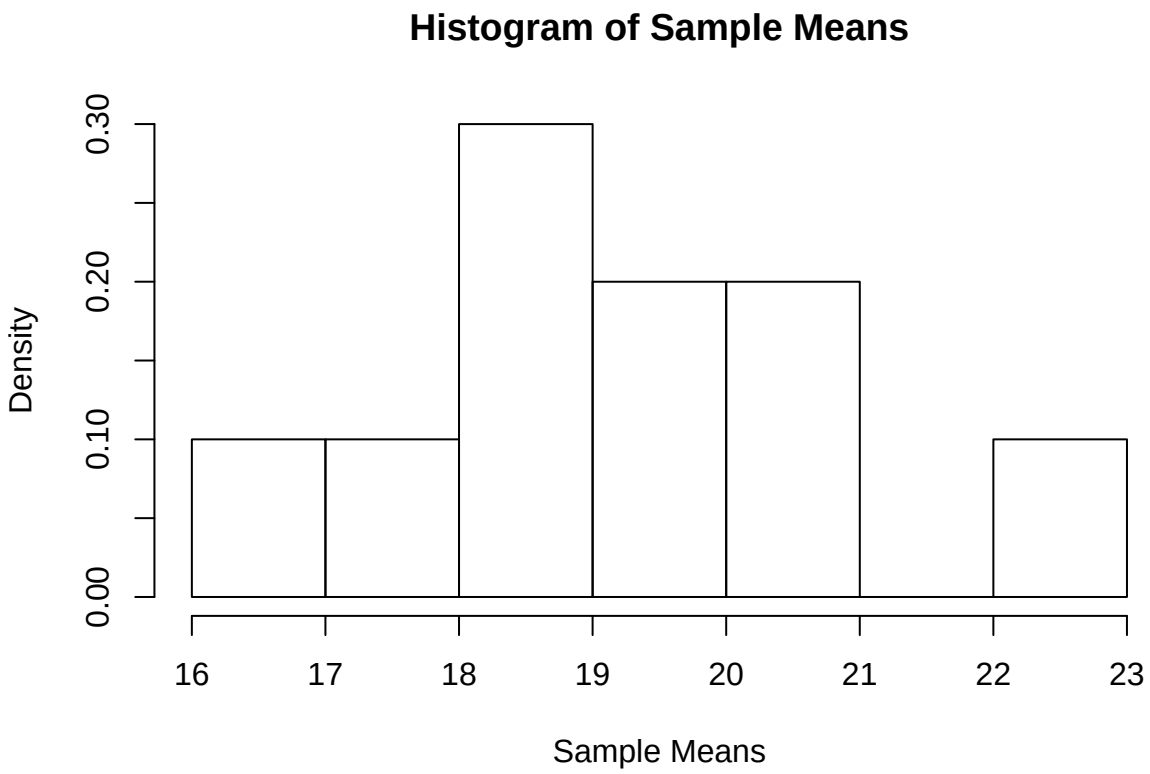
- c.) Write a loop that randomly samples **n times** from your distribution and calculates the sample mean.

```
n <- 10
Result <- sapply(1:n, function(i) {
  # Randomly samples vector.length observations from a Poisson distribution with parameter,
  # then calculates the sample mean
  poisson.sample <- rpois(vector.length, parameter)
  sample.mean <- mean(poisson.sample)
  return(sample.mean)
})
Result
```

```
##  [1] 18.3 22.5 16.6 18.8 18.9 17.7 20.6 19.7 20.0 20.7
```

- d.) Plot a histogram of your sample means.

```r
hist(Result, prob = TRUE, main = "Histogram of Sample Means", xlab = "Sample Means")
```



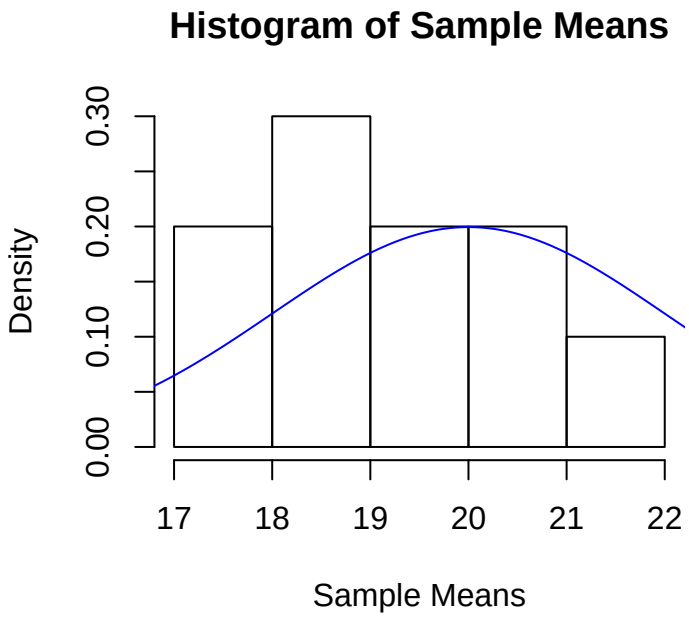
- e.) Hopefully you have written a function that does all of the above for you. Now vary the values of **m** and **n** and compare the resulting plots:
  - m = 5, n = 10
  - m = 5, n = 30
  - m = 5, n = 100
  - m = 15, n = 20
  - m = 15, n = 50
  - m = 15, n = 100
  - m = 50, n = 10
  - m = 50, n = 100
  - m = 50, n = 1000

Optional: Plot a normal density curve around your histogram. What do you notice?

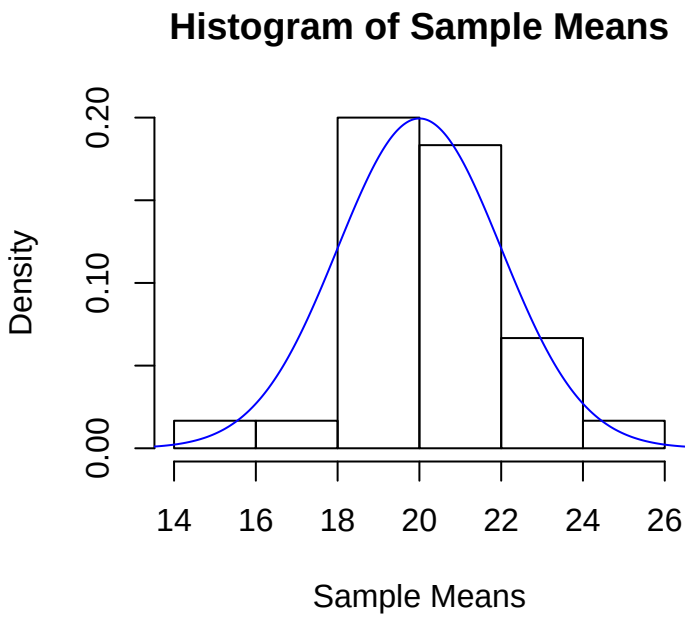
```r
pois.sample <- function(n, vector.length, parameter) {
  # Samples from a Poisson distribution, plots the sample means, then overlays a normal distribution.
  #
  # Args:
  #   n: number of samples desired
```

```
  #   vector.length: number of observations per sample
  #   parameter: desired parameter of the Poisson distribution
  #
  # Returns:
  #   Well, nothing. This function plots.
  Result = sapply(1:n, function(i) {
    poisson.sample <- rpois(vector.length, parameter)
    sample.mean <- mean(poisson.sample)
    return(sample.mean)
  })
  hist(Result, prob = TRUE, main = "Histogram of Sample Means", xlab = "Sample Means")
  x <- seq(parameter - 3 * sd(poisson.sample), parameter + 3 * sd(poisson.sample), length = 1000)
  y <- dnorm(x, mean = parameter, sd = sqrt(parameter) / sqrt(vector.length))
  lines(x, y, col = "blue")
}
```

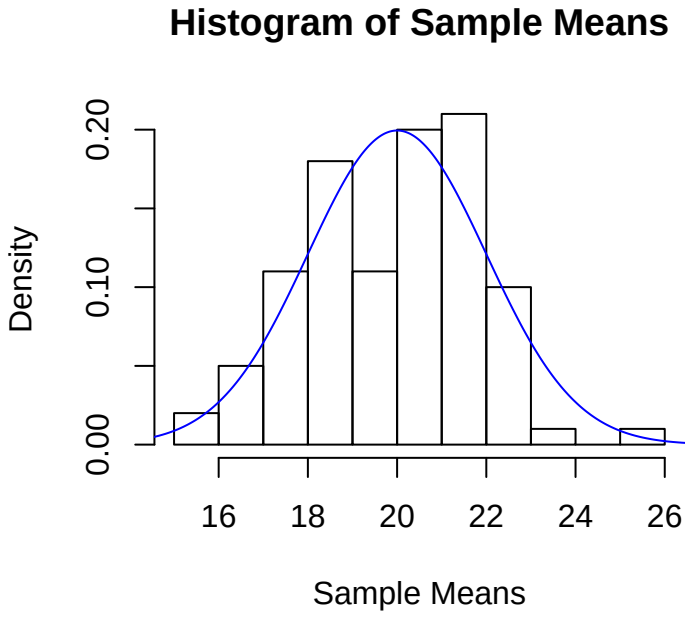
```
pois.sample(n = 10, vector.length = 5, parameter = 20)
```



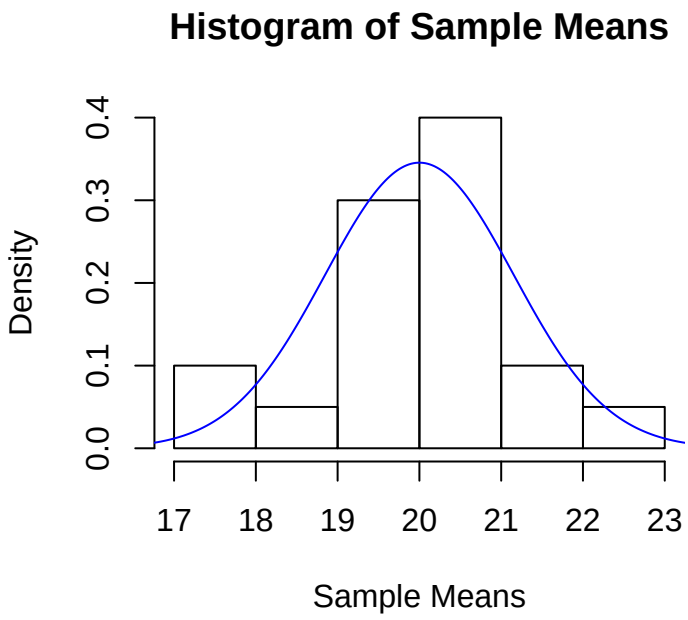

```
pois.sample(n = 30, vector.length = 5, parameter = 20)
```



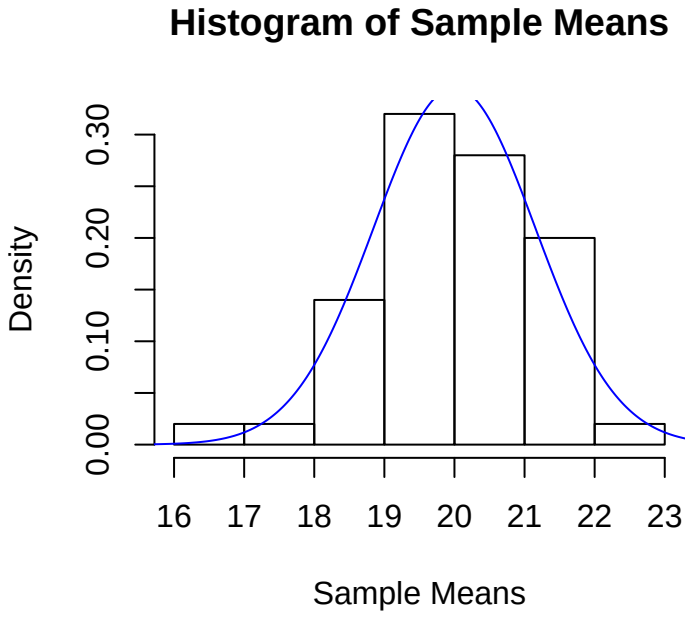

```
pois.sample(n = 100, vector.length = 5, parameter = 20)
```



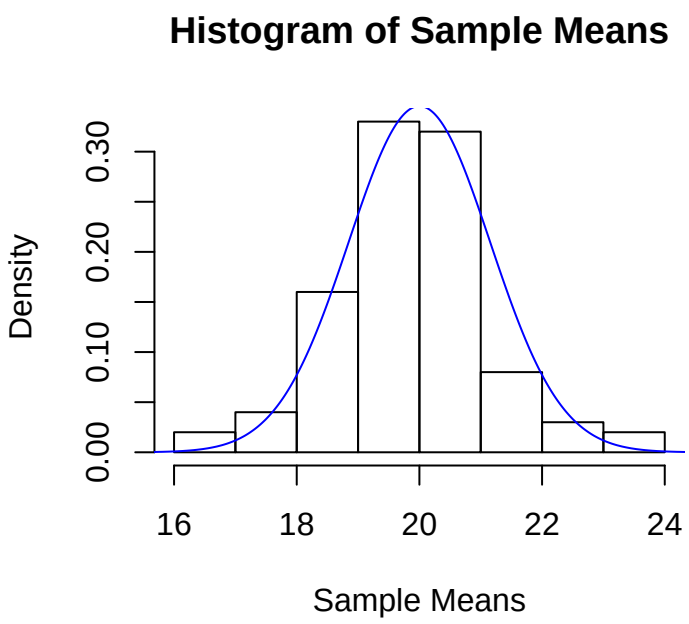

```
pois.sample(n = 20, vector.length = 15, parameter = 20)
```



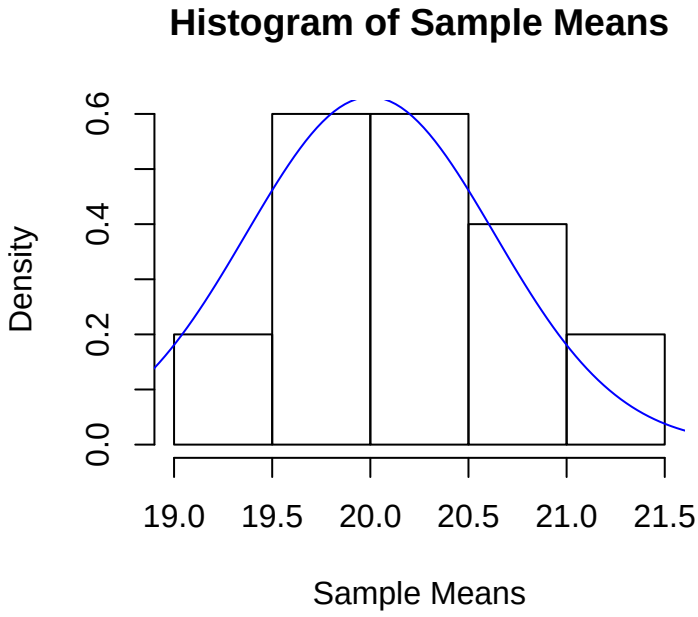

```
pois.sample(n = 50, vector.length = 15, parameter = 20)
```



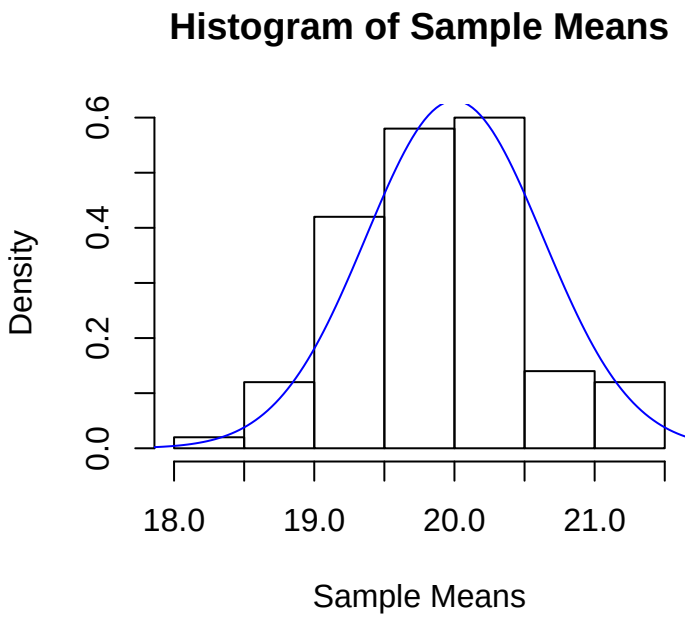

```
pois.sample(n = 100, vector.length = 15, parameter = 20)
```



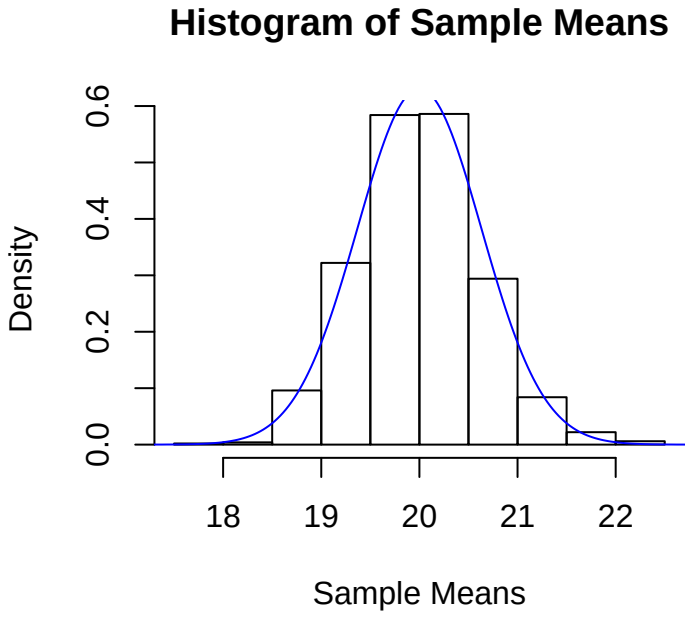

```
pois.sample(n = 10, vector.length = 50, parameter = 20)
```




```
pois.sample(n = 100, vector.length = 50, parameter = 20)
```




```
pois.sample(n = 1000, vector.length = 50, parameter = 20)
```



Exercise 2

What is a confidence interval graphically

We will graphically show what confidence actually is (using $\alpha = 0.05$)

- a.) Take samples of size **m** from a normal distribution (you choose the mean and standard deviation as well as well).

```
# I will use a normal distribution with mean 5 and standard deviation 2.
alpha <- 0.05
vector.length <- 10
true.mean <- 5
standard.deviation <- 2
sample.vector <- rnorm(vector.length, true.mean, standard.deviation)
sample.vector
```

```
##  [1] 6.914133 5.054364 5.217630 6.010910 3.902954 4.477776 7.188629
##  [8] 4.200670 4.318113 3.313961
```

- b.) Calculate the error of the confidence interval.

$Error = Z_{1-\frac{\alpha}{2}} \times \frac{\sigma}{\sqrt{m}}$

```
error <- qnorm(1 - (alpha / 2)) * (sd(sample.vector) / sqrt(length(sample.vector)))
error
```

```
## [1] 0.7978339
```

- c.) Calculate the confidence interval.

$ConfidenceInteral = \bar{X} \pm Z_{1-\frac{\alpha}{2}} \times \frac{\sigma}{\sqrt{m}}$

```
CI <- c(mean(sample.vector) - error, mean(sample.vector) + error)
CI
```

```
## [1] 4.262080 5.857748
```

- d.) Write a loop that randomly samples **n times** from the distribution and calculates the confidence intervals.

```
n <- 10
Result <- sapply(1:n, function(i) {
  # Creates your confidence intervals (n of them)
  sample.vector <- rnorm(vector.length, true.mean, standard.deviation)
  error <- qnorm(1 - (alpha / 2)) * (sd(sample.vector) / sqrt(length(sample.vector)))
  CI <- c(mean(sample.vector) - error, mean(sample.vector) + error)
  return(CI)
})
Result
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## [1,] 2.568873 4.202523 4.059378 3.061893 3.617425 4.279371 3.734983
## [2,] 4.424778 5.290996 7.131118 5.885925 6.389481 6.815858 6.350563
##          [,8]     [,9]    [,10]
## [1,] 4.749121 2.808518 4.917716
## [2,] 6.254909 5.218432 7.396007
```

- e.) Combine everything into one function that calculates confidence intervals from 4 variables: *n, vector.length, true.mean, standard.deviation.*

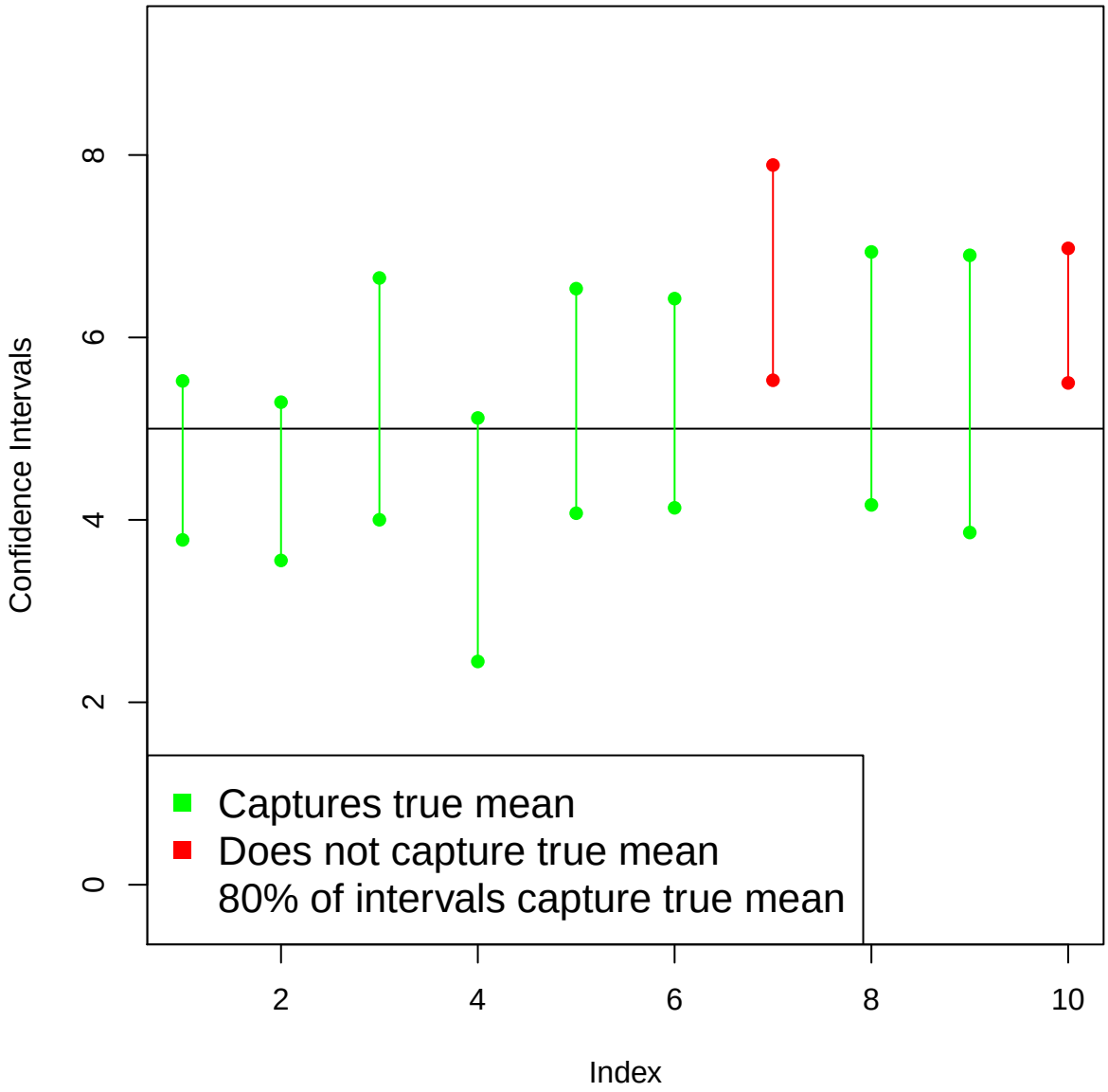
```
ConfIntervals <- function(n, vector.length, true.mean, standard.deviation) {
  # Calculates confidence intervals from a normal distribution given parameters
  #
  # Args:
  #   n: number of confidence intervals desired
  #   vector.length: number of observations per confidence interval
  #   true.mean: true mean of the normal distribution
  #   standard.deviation: true standard deviation of the normal distribution
  #
  # Returns:
  #   A 2 x n data frame of confidence interval bounds.
  Result <- sapply(1:n, function(i) {
    sample.vector <- rnorm(vector.length, true.mean, standard.deviation)
    error <- qnorm(1 - (alpha / 2)) * (sd(sample.vector) / sqrt(length(sample.vector)))
    CI <- c(mean(sample.vector) - error, mean(sample.vector) + error)
    return(CI)
  })
  return(Result)
}
```

- f.) Use the function provided below which takes in the data frame output your confidence interval function. Does it match your knowledge of what confidence intervals are?

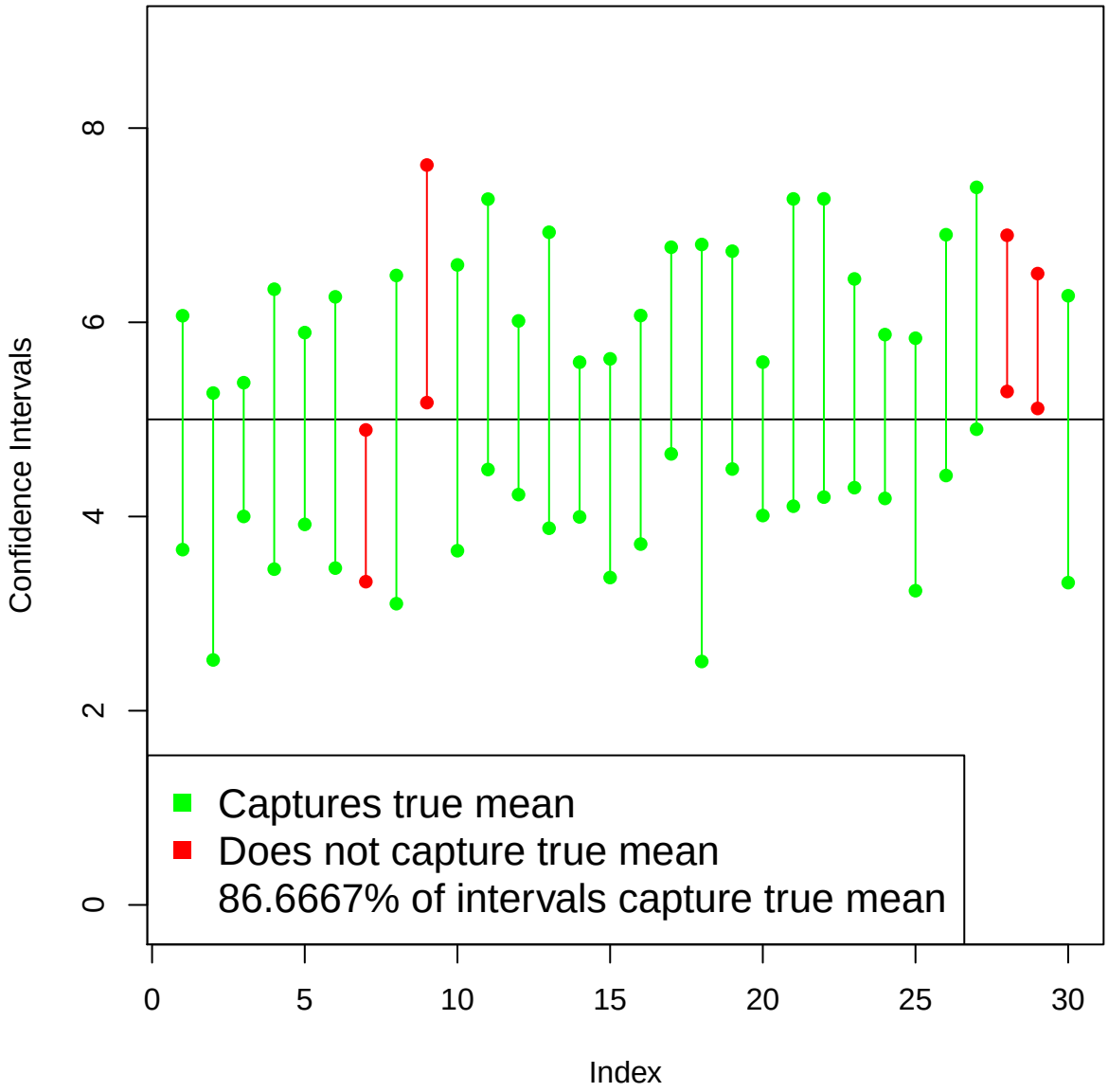
```
PlotIntervals <- function(data) {
  # Plots the input 2 x n confidence interval data frame
  #
  # Args:
  #   data: Data frame of size 2 x n of confidence intervals from ConfIntervals
  #
  # Returns:
  #   A useless vector that will not be seen. This function does plotting.
  lower <- min(data) - (max(data) - min(data)) * 0.5
  upper <- max(data) + (max(data) - min(data)) * 0.25
  plot(1, true.mean, type = "n", xlab = "Index", ylab = "Confidence Intervals",
       xlim = c(1, ncol(data)), ylim = c(lower, upper))
  abline(h = true.mean)
  Result <- sapply(1:ncol(data), function(i) {
    if (data[1, i] <= true.mean && true.mean <= data[2, i]) {
      points(i, data[1, i], col = "green", pch = 16)
      points(i, data[2, i], col = "green", pch = 16)
      segments(x0 = i, y0 = data[1, i], x1 = i, y1 = data[2, i], col = "green")
    } else {
      points(i, data[1, i], col = "red", pch = 16)
```

```
      points(i, data[2, i], col = "red", pch = 16)
      segments(x0 = i, y0 = data[1, i], x1 = i, y1 = data[2, i], col = "red")
    }
    return("")
  })
  Result2 <- sapply(1:ncol(data), function(i) {
    prop.vector = data[1, i] <= true.mean && true.mean <= data[2, i]
    return(prop.vector)
  })
  prop.captured <- paste0(signif((mean(Result2) * 100), digits = 6),
                    "% of intervals capture true mean")
  legend(x = "bottomleft", legend = c("Captures true mean",
        "Does not capture true mean", prop.captured), pch = 15,
        col = c("green", "red", "transparent"), cex = 1.3)
}
```

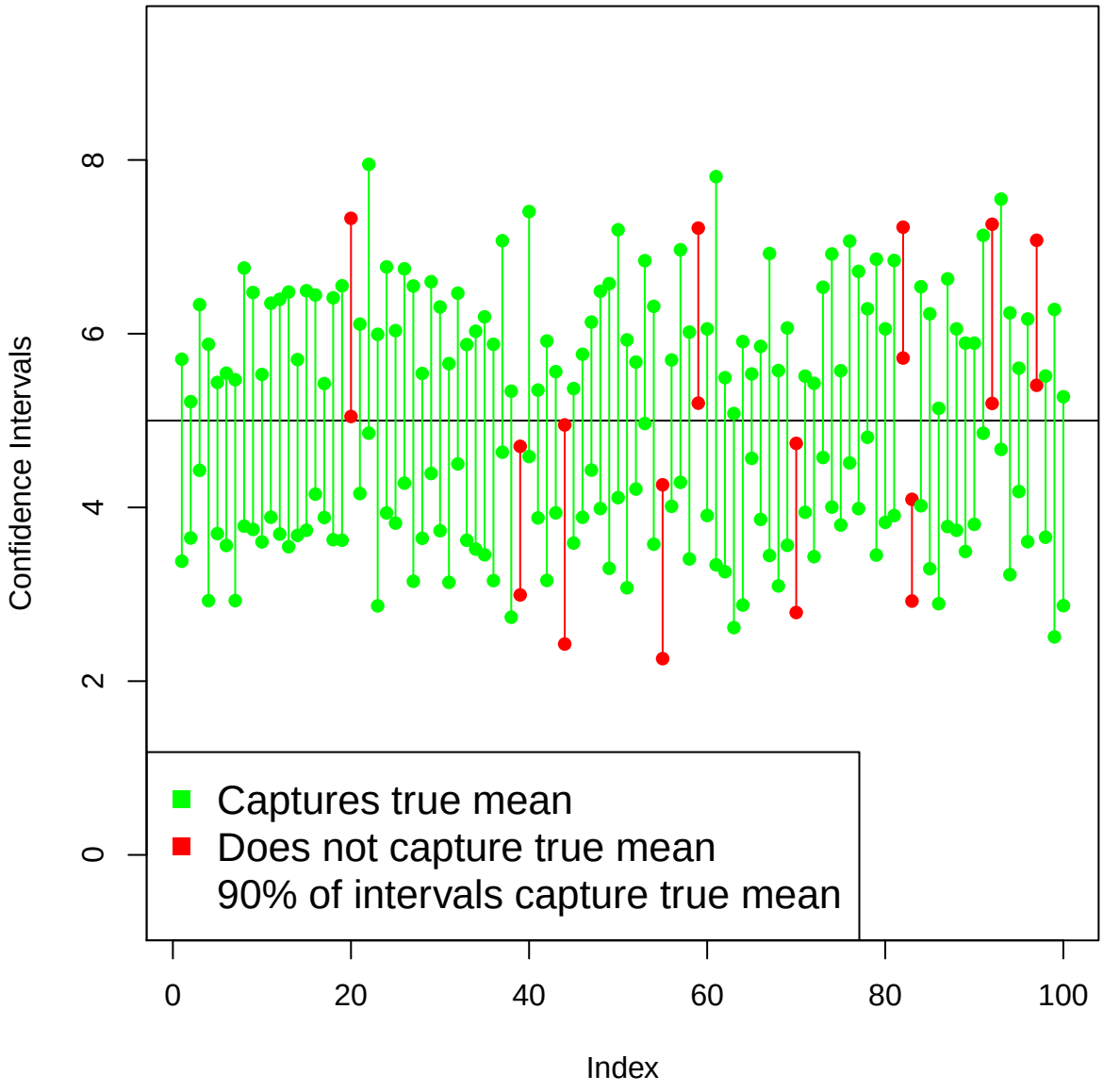
```
PlotIntervals(ConfIntervals(n = 10, vector.length = 10, true.mean = 5, standard.deviation = 2))
```



```
PlotIntervals(ConfIntervals(n = 30, vector.length = 10, true.mean = 5, standard.deviation = 2))
```



```
PlotIntervals(ConfIntervals(n = 100, vector.length = 10, true.mean = 5, standard.deviation = 2))
```



```
PlotIntervals(ConfIntervals(n = 200, vector.length = 10, true.mean = 5, standard.deviation = 2))
```