HW3

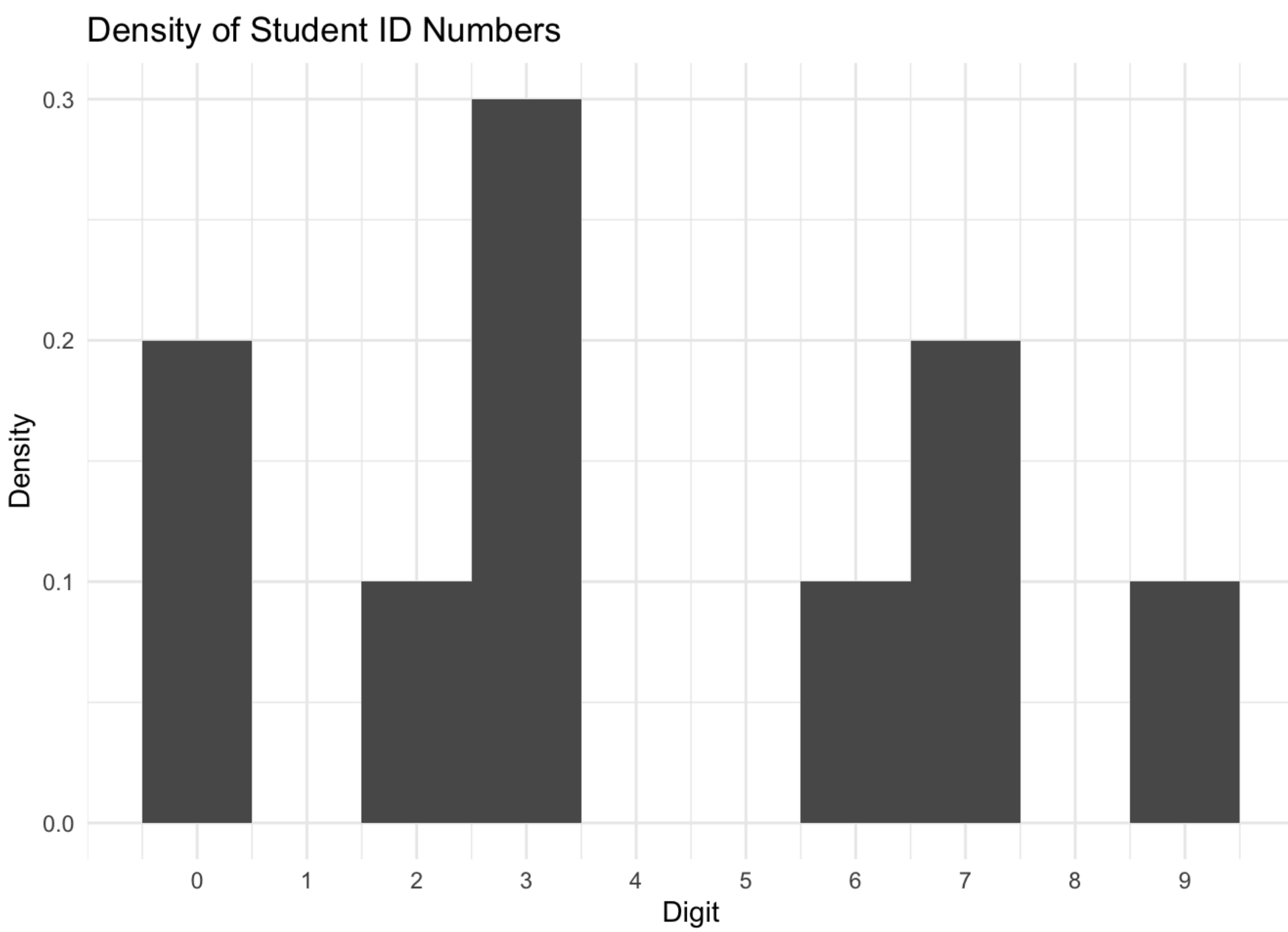
Natalie Brewer

2023-09-12

Problem 3F

```
digits = c(3,0,3,6,3,7,9,0,7,2)
SID <- data.frame(digits)

ggplot(SID, aes(x=digits)) +
  geom_histogram(aes(y=after_stat(density)), binwidth = 1) +
  labs(title="Density of Student ID Numbers", x="Digit", y="Density") +
  scale_x_continuous(breaks=seq(min(digits), max(digits), by=1)) +
  theme_minimal()
```




```
#Find mean
mean <- mean(digits)
mean
```

```
## [1] 4
```

```
#Find standard deviation
sd <- sd(digits)
sd
```

```
## [1] 3.091206
```

Problem 3G

```
sample <- sample(digits, 400, replace=TRUE)
sample
```

```
##   [1] 7 7 3 7 3 3 3 3 3 7 7 2 3 9 2 0 7 3 7 7 7 0 3 0 3 2 3 3 0 7 2 3 3 3 2 3 6
##  [38] 3 3 3 7 3 0 7 0 0 7 0 0 7 0 7 3 0 3 0 7 7 7 3 7 9 2 3 9 2 9 3 7 0 7 0 0 6
##  [75] 0 3 9 6 0 9 3 6 2 3 3 3 3 3 6 3 3 3 0 2 0 3 2 3 3 6 0 7 7 3 3 3 7 0 2 3 7
## [112] 0 7 0 3 2 3 7 2 7 3 3 7 3 0 3 6 7 9 3 3 2 3 7 2 0 0 0 3 3 6 7 7 6 6 7 3 3
## [149] 2 0 0 6 7 9 3 2 9 7 2 0 6 3 6 9 0 3 3 6 7 0 9 3 3 6 6 6 7 7 6 2 6 7 6 3 7
## [186] 9 3 7 3 9 6 2 0 3 9 7 9 7 9 3 9 3 3 7 0 7 7 6 3 0 7 9 7 2 3 0 3 6 7 3 7 7
## [223] 0 3 9 3 3 9 3 3 3 3 7 7 9 9 3 3 3 2 9 3 7 3 0 0 0 9 7 7 0 0 3 0 3 3 9 3 0
## [260] 9 0 3 7 7 0 2 7 7 3 3 9 7 3 3 9 0 3 0 0 6 7 0 9 0 6 6 9 3 3 3 0 0 3 7 3 7
## [297] 2 0 3 0 6 2 3 0 7 3 6 3 2 3 3 6 7 0 2 6 3 2 6 3 3 7 6 0 0 0 3 9 3 3 6 3 7
## [334] 2 3 9 0 9 0 9 6 3 7 6 6 7 3 6 3 2 9 9 6 0 0 3 7 3 3 3 9 0 0 0 3 9 7 0 6 2
## [371] 7 0 3 7 3 0 7 0 3 0 9 3 7 0 9 6 9 3 9 6 3 0 7 2 7 0 7 2 9 0
```

```
sampleMean <- mean(sample)
sampleMean
```

```
## [1] 4.1
```

```
standardError <- sd/sqrt(400)
standardError
```

```
## [1] 0.1545603
```

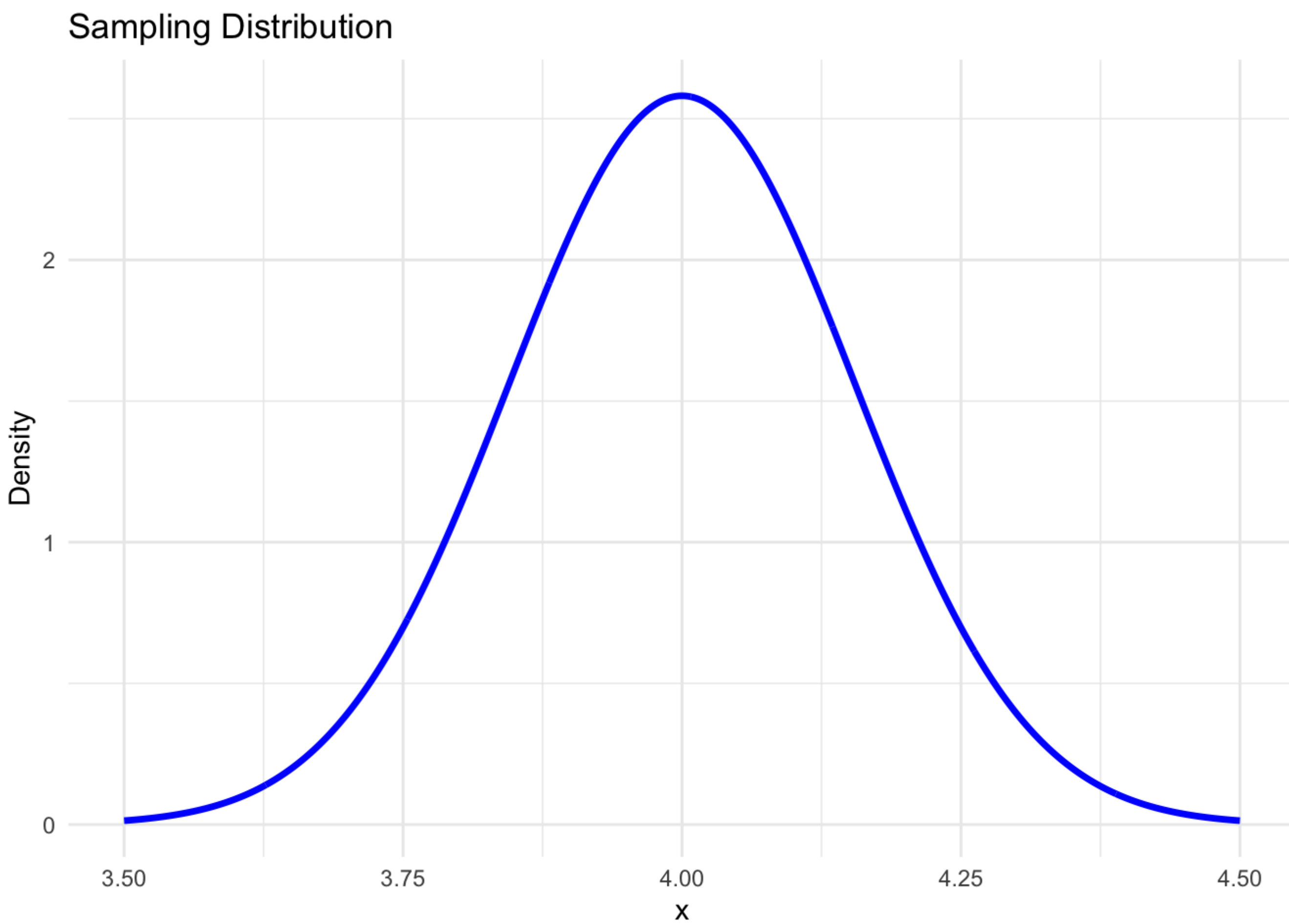
The mean of the sampling distribution is the population mean, 4.

The standard error is

$$\frac{\sqrt{\sigma^2}}{\sqrt{n}} = \frac{3.0912}{\sqrt{400}} = 0.155$$

```
x_vals <- seq(3.5, 4.5, by=0.001)
df <- data.frame(x=x_vals, y=dnorm(x_vals, mean=mean, sd=standardError))

ggplot(df, aes(x=x, y=y)) +
  geom_line(color="blue", linewidth=1.2) +
  labs(title="Sampling Distribution", x="x", y="Density") +
  theme_minimal()
```



Problem 3H

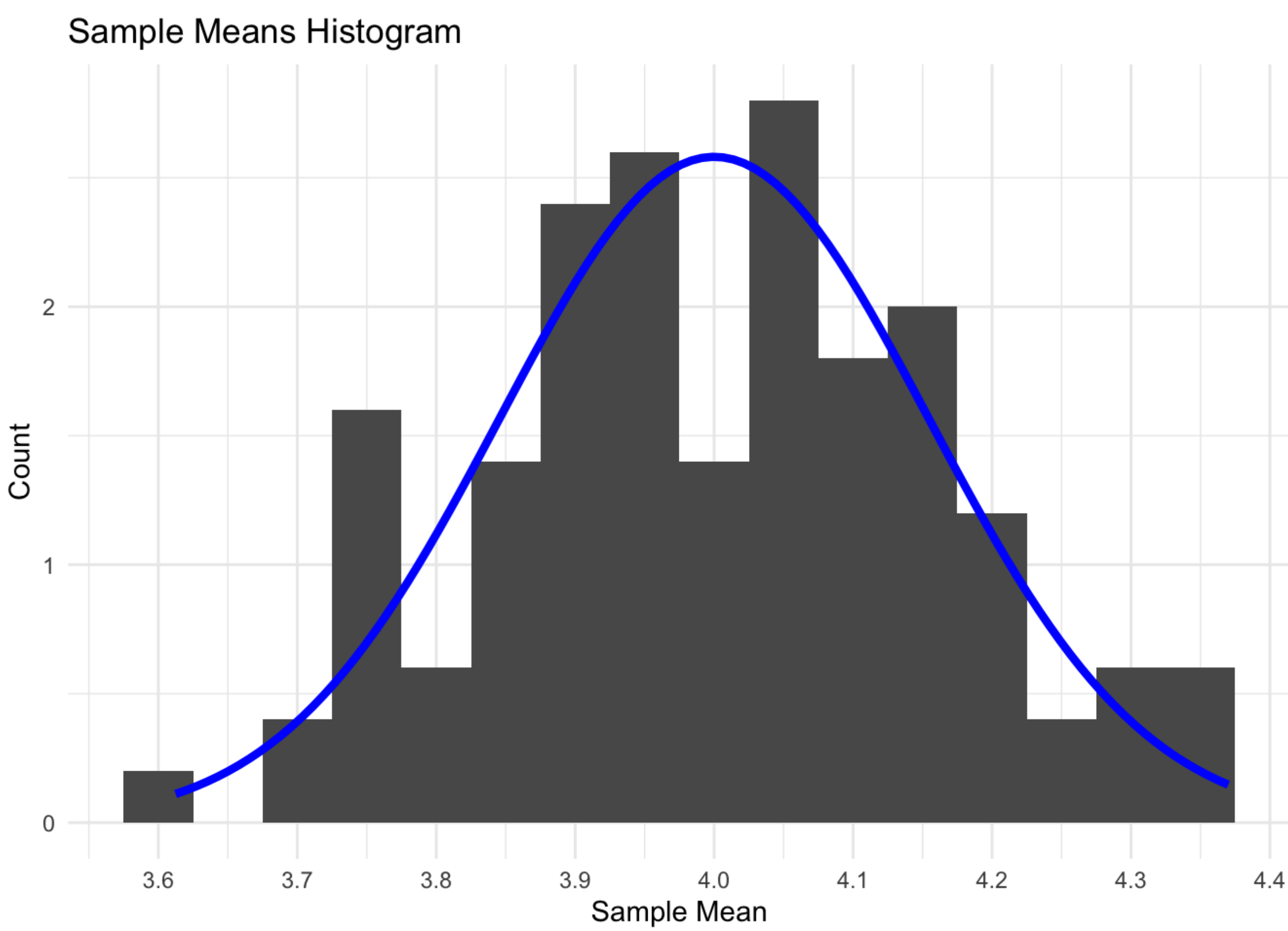
```
find_avg <- function(){
  resample <- sample(digits, size = 400, replace = T)
  xbar <- mean(resample)
  xbar
}

sampleMeanVec <- replicate(100, find_avg())
sampleMeanVec
```

```
##   [1] 4.3125 3.9900 4.0600 4.3450 3.9275 4.1525 4.3200 3.9275 3.9700 3.7475
##  [11] 4.0025 3.9475 4.2100 4.1225 4.1575 3.8775 4.1375 4.1175 4.0750 3.7225
##  [21] 3.7525 4.0900 4.1300 3.9875 3.7650 3.9100 3.8625 3.8600 4.0550 4.3025
##  [31] 3.9950 4.1125 3.9250 3.9100 3.8700 3.9450 4.2550 3.7575 4.1850 3.9650
##  [41] 4.0925 4.0650 3.9700 3.8625 4.0525 4.1200 4.1425 3.8050 3.8850 4.1975
##  [51] 4.1275 3.9225 4.0125 4.0725 4.1700 4.0725 3.7025 3.9175 3.6125 4.1975
##  [61] 3.7625 4.1700 3.7625 4.0800 4.1850 3.9525 4.0675 4.0975 3.9575 4.3300
##  [71] 4.0450 3.8950 3.8075 3.9450 3.7675 4.0125 3.8600 4.2025 4.0700 3.8400
##  [81] 3.8850 3.9025 3.9150 4.0300 3.9700 4.3700 3.8225 3.7500 3.8325 4.2675
##  [91] 3.9200 4.0775 4.0400 4.1350 4.0350 4.1500 3.9275 4.0750 3.9300 3.9800
```

```
sampleMeanDf <- data.frame(sampleMeanVec)

ggplot(sampleMeanDf, aes(x=sampleMeanVec)) +
  geom_histogram(aes(y=after_stat(density)), binwidth = .05) +
  labs(title="Sample Means Histogram", x="Sample Mean", y="Count") +
  scale_x_continuous(breaks=seq(3.5, 4.5, by=.1)) +
  stat_function(fun = dnorm, args = list(mean = mean, sd =standardError), color = "blu
e", linewidth = 1.5) +
  theme_minimal()
```



Problem 3I

```
CIs <- data.frame(lower = c(), upper = c())

find_CI <- function(mean, width){
  z <- qnorm(1-((100-width)/200))

  lower <- mean - (z*standardError)
  upper <- mean + (z*standardError)

  return(list(lower = lower, upper = upper))
}

for (i in sampleMeanVec){
  CIs <- rbind(CIs, find_CI(i,95))
}

CIs
```

```
##        lower    upper
## 1   4.009567 4.615433
## 2   3.687067 4.292933
## 3   3.757067 4.362933
## 4   4.042067 4.647933
## 5   3.624567 4.230433
## 6   3.849567 4.455433
## 7   4.017067 4.622933
## 8   3.624567 4.230433
## 9   3.667067 4.272933
## 10  3.444567 4.050433
## 11  3.699567 4.305433
## 12  3.644567 4.250433
## 13  3.907067 4.512933
## 14  3.819567 4.425433
## 15  3.854567 4.460433
## 16  3.574567 4.180433
## 17  3.834567 4.440433
## 18  3.814567 4.420433
## 19  3.772067 4.377933
## 20  3.419567 4.025433
## 21  3.449567 4.055433
## 22  3.787067 4.392933
## 23  3.827067 4.432933
## 24  3.684567 4.290433
## 25  3.462067 4.067933
## 26  3.607067 4.212933
## 27  3.559567 4.165433
## 28  3.557067 4.162933
## 29  3.752067 4.357933
## 30  3.999567 4.605433
## 31  3.692067 4.297933
## 32  3.809567 4.415433
## 33  3.622067 4.227933
## 34  3.607067 4.212933
## 35  3.567067 4.172933
## 36  3.642067 4.247933
## 37  3.952067 4.557933
## 38  3.454567 4.060433
## 39  3.882067 4.487933
## 40  3.662067 4.267933
## 41  3.789567 4.395433
## 42  3.762067 4.367933
## 43  3.667067 4.272933
## 44  3.559567 4.165433
## 45  3.749567 4.355433
## 46  3.817067 4.422933
## 47  3.839567 4.445433
## 48  3.502067 4.107933
## 49  3.582067 4.187933
## 50  3.894567 4.500433
## 51  3.824567 4.430433
```

```
## 52  3.619567 4.225433
## 53  3.709567 4.315433
## 54  3.769567 4.375433
## 55  3.867067 4.472933
## 56  3.769567 4.375433
## 57  3.399567 4.005433
## 58  3.614567 4.220433
## 59  3.309567 3.915433
## 60  3.894567 4.500433
## 61  3.459567 4.065433
## 62  3.867067 4.472933
## 63  3.459567 4.065433
## 64  3.777067 4.382933
## 65  3.882067 4.487933
## 66  3.649567 4.255433
## 67  3.764567 4.370433
## 68  3.794567 4.400433
## 69  3.654567 4.260433
## 70  4.027067 4.632933
## 71  3.742067 4.347933
## 72  3.592067 4.197933
## 73  3.504567 4.110433
## 74  3.642067 4.247933
## 75  3.464567 4.070433
## 76  3.709567 4.315433
## 77  3.557067 4.162933
## 78  3.899567 4.505433
## 79  3.767067 4.372933
## 80  3.537067 4.142933
## 81  3.582067 4.187933
## 82  3.599567 4.205433
## 83  3.612067 4.217933
## 84  3.727067 4.332933
## 85  3.667067 4.272933
## 86  4.067067 4.672933
## 87  3.519567 4.125433
## 88  3.447067 4.052933
## 89  3.529567 4.135433
## 90  3.964567 4.570433
## 91  3.617067 4.222933
## 92  3.774567 4.380433
## 93  3.737067 4.342933
## 94  3.832067 4.437933
## 95  3.732067 4.337933
## 96  3.847067 4.452933
## 97  3.624567 4.230433
## 98  3.772067 4.377933
## 99  3.627067 4.232933
## 100 3.677067 4.282933
```

```
count <- 0  #this will be the number of intervals containing the actual population mean

for (i in 1:nrow(CIs)){
  lower <- CIs[i,1]
  upper <- CIs[i,2]
  if((mean > lower) && (mean < upper)){
    count <- count + 1
  }
}
count
```

```
## [1] 94
```

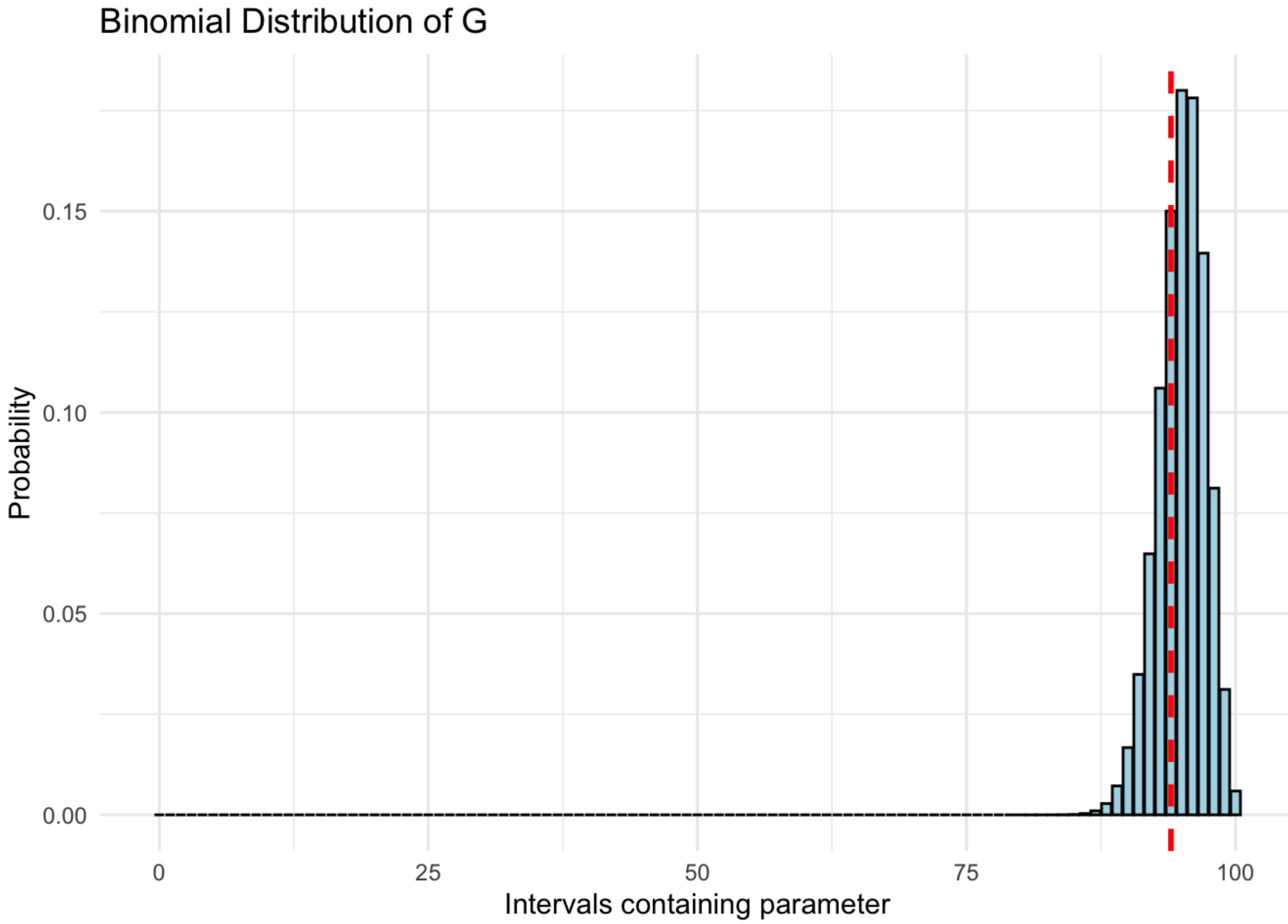
Problem 3J

The probability that each interval contains the actual parameter is 95%. This is a Bernouli trial with p = .95. When we take 100 samples and look at the confidence interval determined by each, the distribution is Binomial(100, .95). E(G) = np = 95 SE(G) = np(1-p) = 4.75

```
df <- data.frame(x = 0:100, probabilities = dbinom(0:100, size = 100, prob = 0.95))

ggplot(df, aes(x = x, y = probabilities)) +
  geom_col(fill = "lightblue", color = "black") +
  geom_vline(aes(xintercept = count), color = "red", linetype = "dashed", size = 1) +
  labs(title = "Binomial Distribution of G", x = "Intervals containing parameter", y =
"Probability") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Based on my values for E(G) and SE(G), the placement of the vertical line from part 3I makes sense. It is close to the mean.