Economics 430 Homework1

Anshika Sharma (ID: 305488635)

12 October 2020

Contents

1. Evans & Rosenthal: 4.3.13

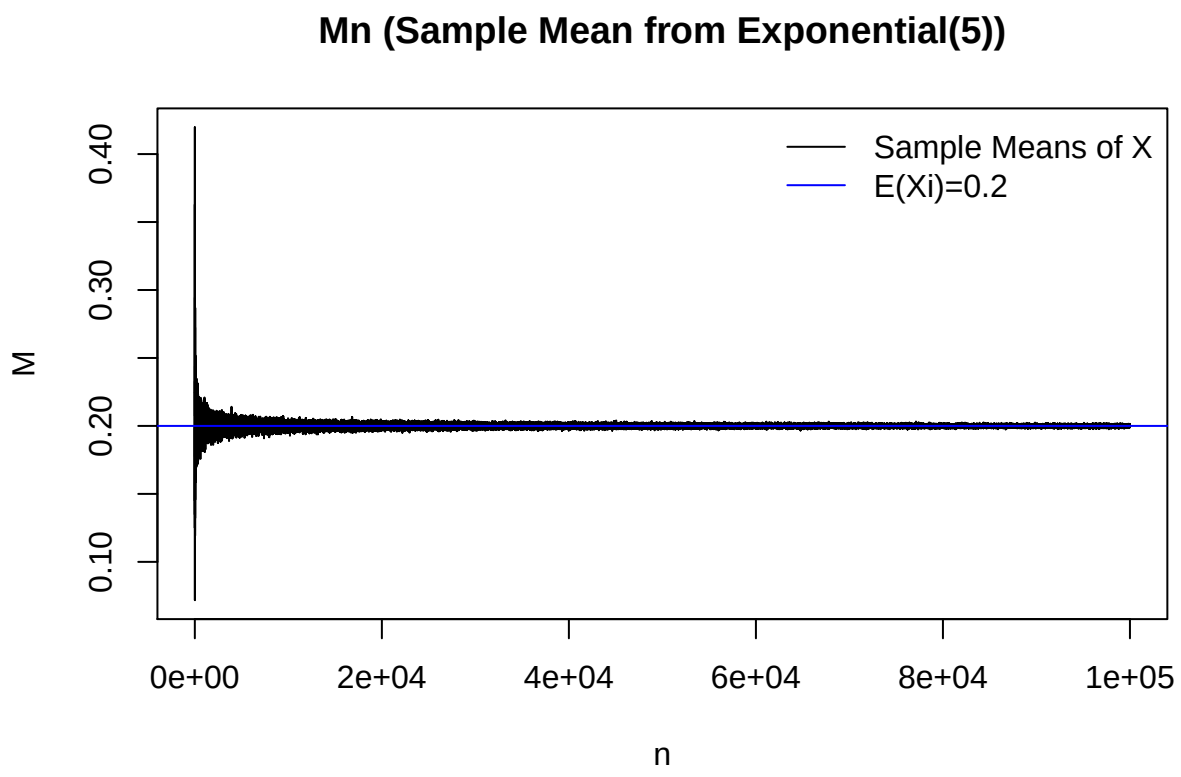
Define $M_n \equiv \sum_{i=1}^{n} X_i/n$. We can create a function that generates the M_n, Mean value of n i.i.d random variables (in this case, $n = 20$) $X_1, X_2, ..., X_n \sim$ Exponential(5) by the following code:

```
generate_Mn<-function(n=20){
  X<-rexp(n, rate=5)
  return(mean(X))
}
```

The following code generates 10^5 samples of$M_{20} \equiv \sum_{i=1}^{20} X_i/20$ which is stored in vector `M`

```
M<-unlist(lapply(1:(10^5), generate_Mn))
```

```
plot(M, type='l', xlab="n", main= "Mn (Sample Mean from Exponential(5))")
abline(h=0.2, col = "blue")
legend("topright",c("Sample Means of X","E(Xi)=0.2"),col =c("black","blue"),
       bty='n',lty = c(1,1))
```



As is shown in the Figure, M_n converges to $E(X_i) = 0.2$ $(\forall i = 1, 2, ..., n)$. This result is consistent with the (strong) law of large numbers.
The figure also implies that M_n converges to 0.2 very quickly. By doing the following computation in R code:

```
max_0.1 <- max((1:length(M))[abs(M-0.2)>=0.1])
print(max_0.1)
```

```
## [1] 6
```

```
max_0.05 <- max((1:length(M))[abs(M-0.2)>=0.05])
print(max_0.05)
```

```
## [1] 44
```

we can see that all the M_n with $n > 6$ ($n > 44$) satisfies $|M_n - 0.2| < 0.1$ ($|M_n - 0.2| < 0.05$).

2. Evans & Rosenthal: 4.4.19

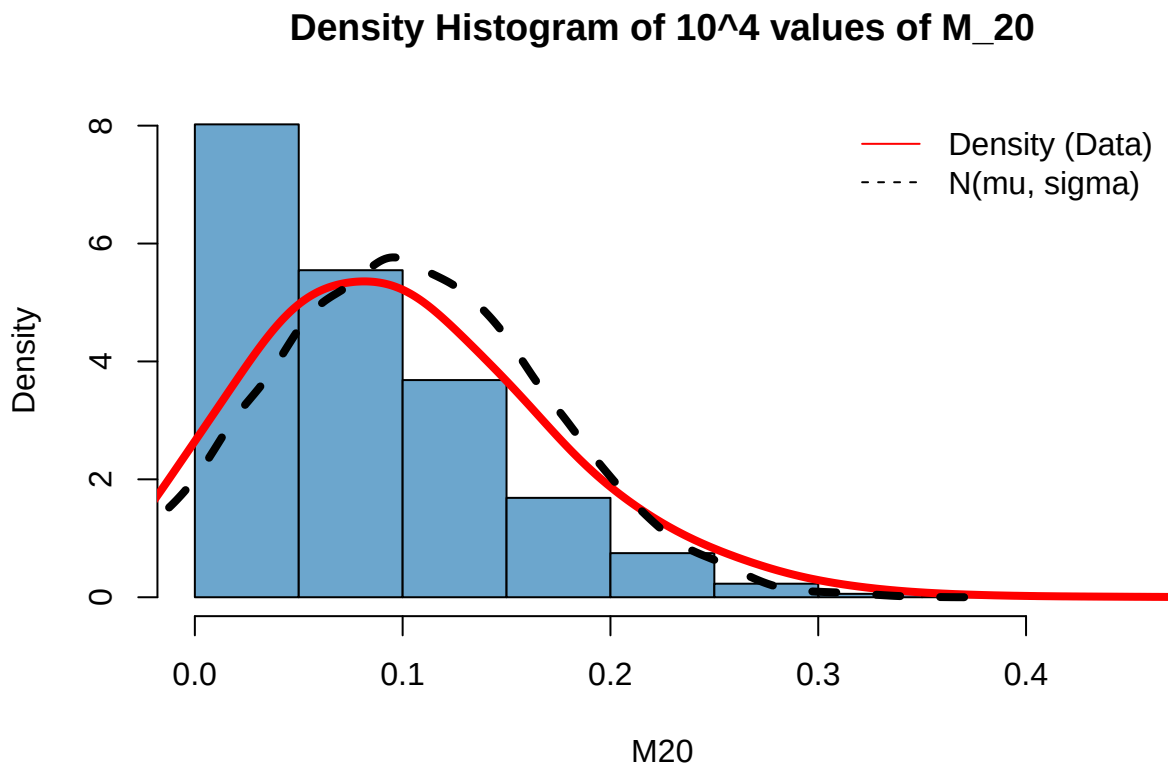
Define $M_n \equiv \sum_{i=1}^{n} X_i/n$. We can create a function that generates the M_n, Mean value of n i.i.d random variables (in this case, $n = 20$) $X_1, X_2, ..., X_n \sim$ Binomial(10,0.1) by the following code:

```
generate_Mn<-function(n=20){
  X<-rbinom(n, 10, 0.01)
  return(mean(X))
}
```

The following code generates 10^4 samples of $M_{20} \equiv \sum_{i=1}^{20} X_i/20$ which is stored in vector `M20`:

```
M20<-replicate(10^4, generate_Mn())
```

```
hist(M20,col="skyblue3",ylab="Density",xlab="M20",prob=TRUE,
     main= "Density Histogram of 10^4 values of M_20")
lines(density(M20, adjust = 3),lwd=4,col='red')
u = rnorm(10^4,mean = mean(M20), sd = sd(M20))
lines(density(u),lwd=4,col='black',lty=2)
legend("topright",c("Density (Data)","N(mu, sigma)"),col =c("red","black"),
       bty='n',lty = c(1,2))
```



We added the density curve (black line) of $N(\mu, \sigma)$ with $\mu = E(M_{20}) = 0.1$ and $\sigma = \sqrt{Var(M_{20})} \simeq 0.0704$ in figure in order to compare the density curve of histogram of M_{20} with a normal density curve.

The figure shows that the histogram is not symmetrical but somewhat skewed to the right or in other words, positively skewed and it does not fit the normal distribution . This is because $n = 20$ and hence is very small. If we choose a larger n, then the distribution of the means will be more symmetric and will be a good approximation to the normal distribution and sample mean will converge to the true mean.

3. Evans & Rosenthal: 4.5.15

Let $I \equiv \int_0^\infty 25cos(x^4)e^{-25x}dx$ and X be a random variable following Exponential(25). Then we have

$$E(cos(X^4)) = \int_0^\infty 25cos(x^4)e^{-25x}dx = I.$$

Hence, we can estimate I by using a Monte Carlo approximation: for large n ($n = 10^5$), I is approximately equal to $M_n = \sum_{i=1}^n t_i/n$, where $t_i = cos(X_i^4)$ with $X_1, ..., X_n$ i.i.d. $\sim$ Exponential(25).

```
X <- rexp(10^5,rate=25)
T <- cos(X^4)
M_n <- mean(T)
print(M_n)
```

```
## [1] 0.9999999
```

From this we can see that I is approximately equal to M_n (almost 1) In order to assess the error in the approximation, we can compute the standard error of Mean values by the following R code:

```
#Compute se of m
S <- sd(T)
se_M_n <- S/sqrt(10^5)
print(se_M_n)
```

```
## [1] 1.677553e-08
```

```
#Compute the interval
L <- M_n-3*se_M_n
U <- M_n+3*se_M_n
print(c(L,U))
```

```
## [1] 0.9999998 0.9999999
```

The standard error of M_n is $S/\sqrt{n}$ where S is a sample standard deviation of $t_i, i = 1, ..., n$. The standard error $S/\sqrt{n}$ is very small and the lower and upper bounds of the confidence interval $(M_n - 3S/\sqrt{n}, M_n + 3S/\sqrt{n})$ are computed and the true value M_n is almost certainly in the interval. Therefore, we can say that the approximation seems to be accurate enough.

We can calculate the actual value of the integral by the following code:

```
I = function(x){
  25*cos(x^4)*exp(-25*x)
}
true_int_value <- integrate(I,0,Inf)
names(true_int_value)
```

```
## [1] "value"        "abs.error"    "subdivisions" "message"      "call"
```

```
print(true_int_value$value)
```

```
## [1] 0.9999999
```

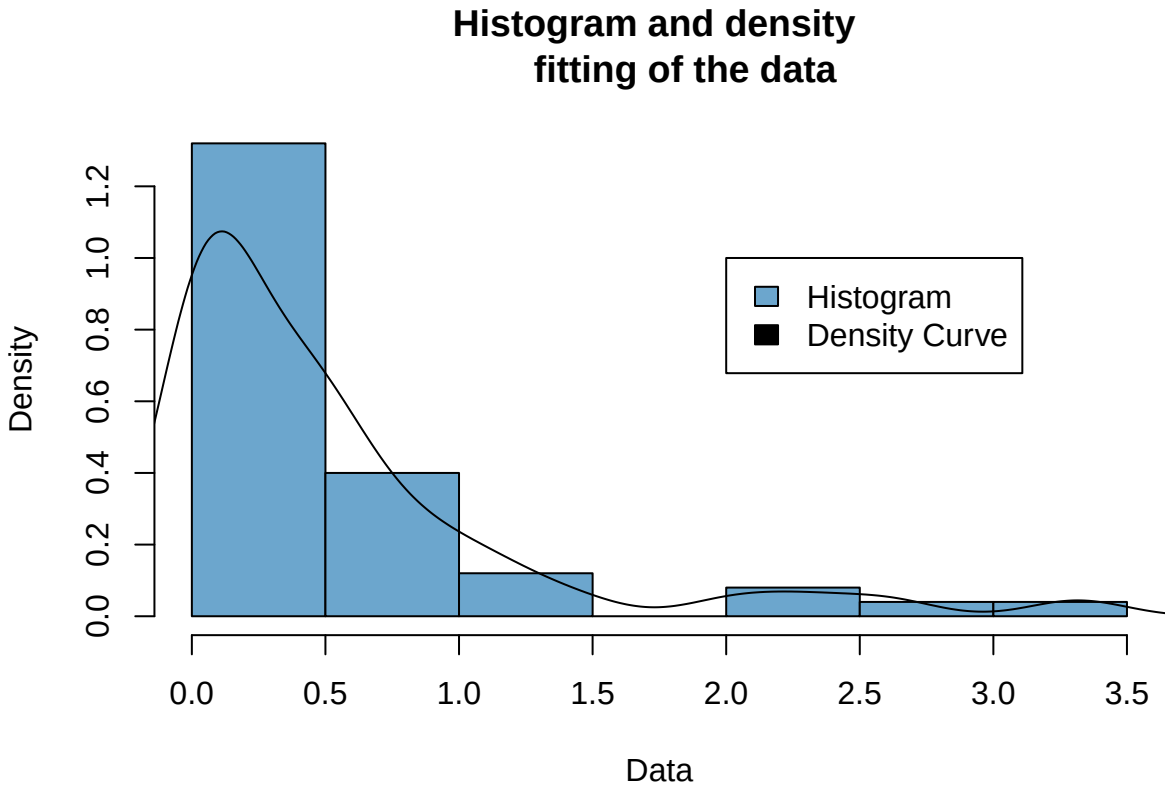
Note that the true value of the Integral is very close to the value we calculated by Monte Carlo approximation.

4. Estimating the Gamma Distribution

(a)

We can plot the histogram of the data ('Prob4 data.txt') and draw the density curve using the following R code:

```
data <- read.table("Prob4_data.txt",col.names="X")
hist(data$X, col = "skyblue3", freq = FALSE, main = "Histogram and density
     fitting of the data",xlab="Data", ylab="Density")
lines(density(data$X), col = "black")
legend(2.0,1.0,c("Histogram","Density Curve"),fill=c("skyblue3","black"))
```



(b)

Let X_i $(i = 1, 2, ..., 50)$ be the values of the data. Suppose a random variable X has the Gamma(α, β) distribution. Then the pdf of X is given by

$$f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}.$$

when $x > 0$ with $f(x) = 0$ for $x \leq 0$. So, the first and second moments of X, $\mu_i \equiv E(X^i)$ $(i = 1, 2)$ can be computed as follows:

$$\begin{aligned}
\mu_i &= \int_0^\infty x^i f(x) dx \\
&= \frac{\beta^\alpha}{\Gamma(\alpha)} \int_0^\infty x^{\alpha+i-1} e^{-\beta x} dx \\
&= \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{\Gamma(\alpha+i)}{\beta^{\alpha+i}} \\
&= \begin{cases} \frac{\alpha}{\beta} & (i = 1) \\ \frac{\alpha(\alpha+1)}{\beta^2} & (i = 2) \end{cases}
\end{aligned}$$

So we get

$$\begin{aligned}
\alpha &= \frac{\mu_1^2}{\mu_2 - \mu_1^2}, \\
\beta &= \frac{\mu_1}{\mu_2 - \mu_1^2}.
\end{aligned}$$

Using the Method of Moments, we can obtain the estimates of the parameters, $\hat{\alpha}$ and $\hat{\beta}$, as follows:

$$\hat{\alpha} = \frac{m_1^2}{m_2 - m_1^2},$$
$$\hat{\beta} = \frac{m_1}{m_2 - m_1^2}.$$

where $m_i \equiv \frac{1}{50}\sum_{j=1}^{50} X_j^i$ $(i = 1, 2)$. The corresponding R code is as follows:

```
m_1 <- mean(data$X)
m_2 <- mean(data$X^2)
alphahat <- (m_1^2)/(m_2-m_1^2)
betahat <- (m_1)/(m_2-m_1^2)
print(c(alphahat,betahat))
```

```
## [1] 0.5805419 1.0818491
```

Therefore, we get $\hat{\alpha} = 0.5805419$ and $\hat{\beta} = 1.0818491$.

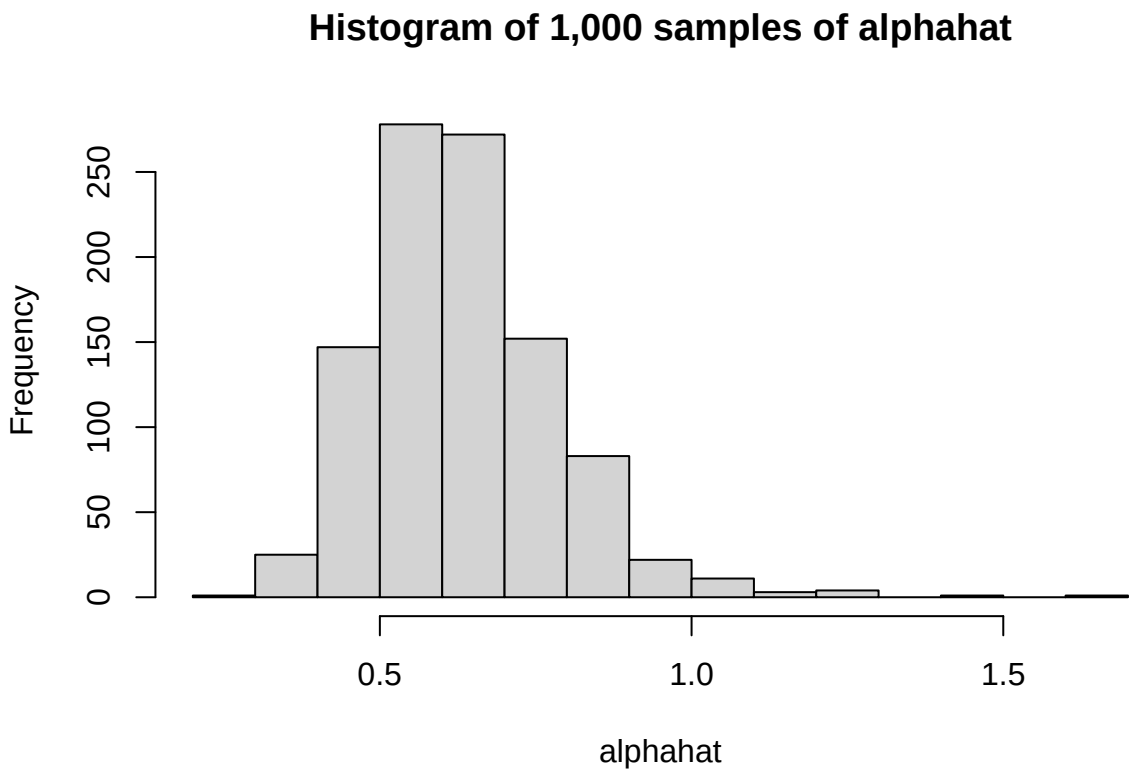
(c)

We can generate 1,000 bootstrap samples by the following R code.

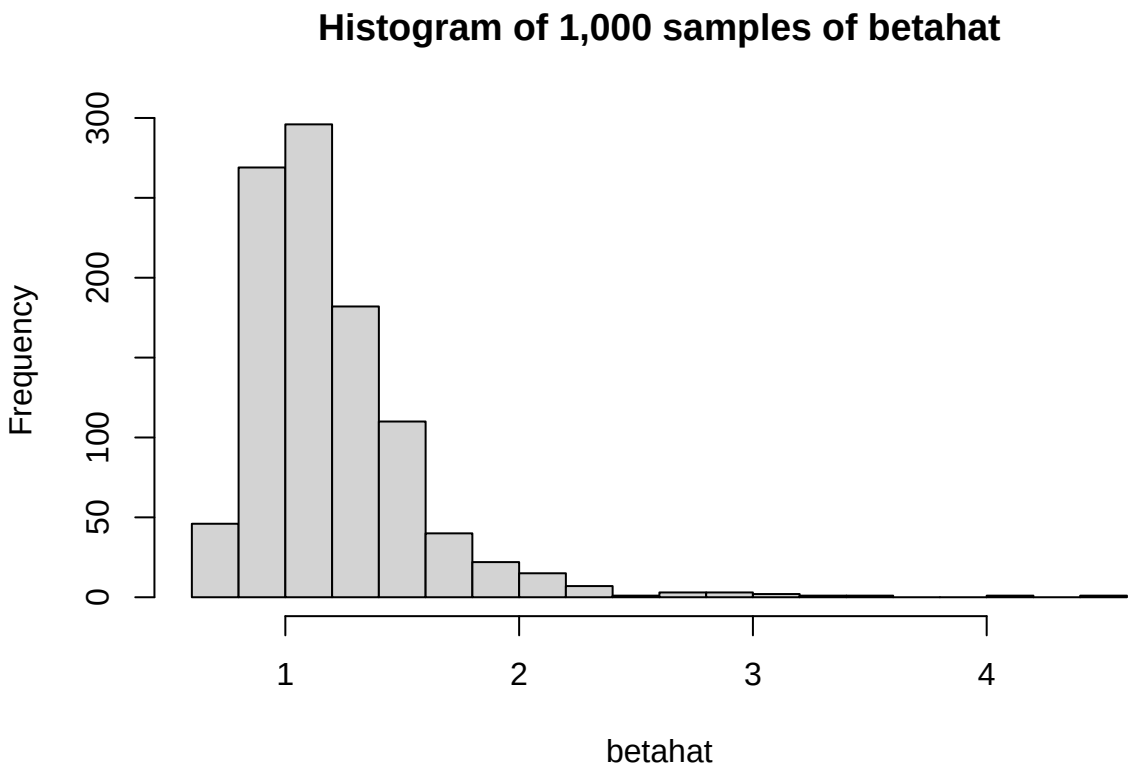
```
set.seed(1234)
N <- 10^3
boot_samples <- NULL
for(i in 1:N){
  sample_i <- sample(data$X,size=length(data$X),replace=TRUE)
  boot_samples <- rbind(boot_samples,sample_i)
}
dimnames(boot_samples) <- NULL
```

Using `boot_samples`, we can generate the sample vectors of $\hat{\alpha}$ (`alphahat_boot`) and $\hat{\beta}$ (`betahat_boot`) as follows:

```
alphahat_boot <- numeric(N)
betahat_boot <- numeric(N)
for(i in 1:N){
  m_1_boot <- mean(boot_samples[i,])
  m_2_boot <- mean(boot_samples[i,]^2)
  alphahat_boot[i] <- (m_1_boot^2)/(m_2_boot-m_1_boot^2)
  betahat_boot[i] <- (m_1_boot)/(m_2_boot-m_1_boot^2)
}
hist(alphahat_boot, main = "Histogram of 1,000 samples of alphahat",
     xlab="alphahat", ylab="Frequency", breaks=15)
```




```
hist(betahat_boot, main = "Histogram of 1,000 samples of betahat",
     xlab="betahat", ylab="Frequency", breaks=15)
```



Hence, we can compute the bootstrap means, standard errors, and 95% confidence intervals of the parameters by the following R code:

```
#Compute the bootstrap mean and se of alpha
mean_alpha_boot <- mean(alphahat_boot)
se_alpha_boot <- sd(alphahat_boot)
print(c(mean_alpha_boot, se_alpha_boot))
```

```
## [1] 0.6344936 0.1486056
```

```
#Compute the bootstrap t 95% CI of alpha
t_ci_alpha_boot <- mean_alpha_boot + qt(c(0.025, 0.975), df=length(data$X)-1)*se_alpha_boot
print(t_ci_alpha_boot)
```

```
## [1] 0.3358595 0.9331277
```

```
#Compute the bootstrap percentile 95% CI of alpha
k <- round(c(0.025,0.975)*N)
p_ci_alpha_boot <- sort(alphahat_boot)[k]
print(p_ci_alpha_boot)
```

```
## [1] 0.3992129 0.9777358
```

```r
#Compute the bootstrap mean and se of beta
mean_beta_boot <- mean(betahat_boot)
se_beta_boot <- sd(betahat_boot)
print(c(mean_beta_boot, se_beta_boot))
```

```
## [1] 1.1993968 0.3738645
```

```r
#Compute the bootstrap t 95% CI of beta
t_ci_beta_boot <- mean_beta_boot + qt(c(0.025, 0.975), df=length(data$X)-1)*se_beta_boot
print(t_ci_beta_boot)
```

```
## [1] 0.448088 1.950706
```

```r
#Compute the bootstrap percentile 95% CI of beta
k <- round(c(0.025,0.975)*N)
p_ci_beta_boot <- sort(betahat_boot)[k]
print(p_ci_beta_boot)
```

```
## [1] 0.7625718 2.1005797
```

Note that it would be appropriate to compute not only bootstrap t CIs (`t_ci_alpha_boot` and `t_ci_beta_boot`) but also bootstrap percentile CIs (`p_ci_alpha_boot` and `p_ci_beta_boot`), since Figures 4 shows that neither $\hat{\alpha}$ nor $\hat{\beta}$ is symmetrically distributed, which implies that we cannot use t statistics as inference. Because the histograms of $\hat{\alpha}$ and $\hat{\beta}$ are skewed to the right, the bounds of percentile CIs are obtained to be greater than those of t CIs.
Summarizing the above, $\overline{\hat{\alpha}} = 0.6344936$, $se(\hat{\alpha}) = 0.1486056$, t 95% CI of $\hat{\alpha} = (0.3358595, 0.9331277)$, percentile 95% CI of $\hat{\alpha} = (0.3992129, 0.9777358)$, $\overline{\hat{\beta}} = 1.1993968$, $se(\hat{\beta}) = 0.3738645$, t 95% CI of $\hat{\beta} = (0.448088, 1.9507056)$, percentile 95% CI of $\hat{\beta} = (0.7625718, 2.1005797)$.
Since both $\hat{\alpha}$ and $\hat{\beta}$ obtained in (b) fall in the respective 95% CIs (both t and percentile CIs), we can say that our results above are compatible with the results in (b).

5. Evans & Rosenthal: 5.4.12

(a)

We can take a simple random sample `srs_sample` by randomly selecting a subset from population, i.e. sampling without replacement:

```r
srs_sample<-sample(1:10000, 500)
```

(b)

We can take an i.i.d. random sample `iid_sample` by randomly and independently selecting each element from the same population, i.e. sampling with replacement:

```r
iid_sample<-sample(1:10000, 500, replace=TRUE)
```

6. Evans & Rosenthal: 5.5.18

(a)

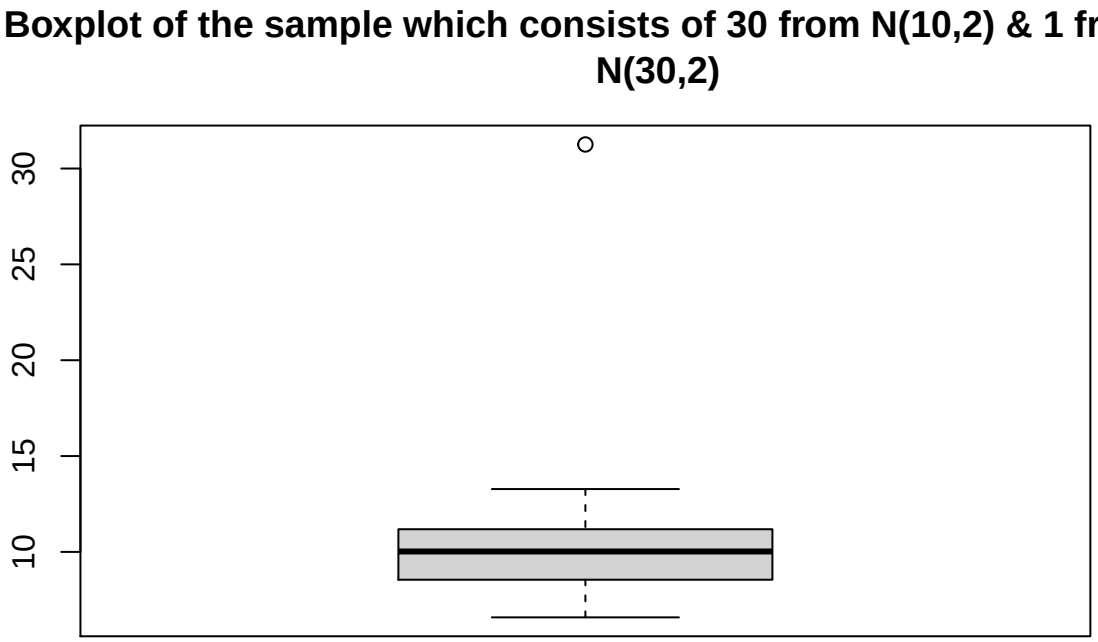
We can generate the sample `sample`, which consists of 30 observations from N(10,2) and 1 observation from N(30,2), by the following R code:

```
sample_1 <- rnorm(30,mean=10,sd=sqrt(2))
sample_2 <- rnorm(1, mean=30,sd=sqrt(2))
sample <- append(sample_1, sample_2)
sample
```

```
##  [1] 10.745511  9.666378  6.868645  6.585504 12.882925 11.110004  7.657489
##  [8]  8.205873 11.035704 10.640604 10.081018 11.490763  7.916555 10.021851
## [15]  9.980773  9.128965 13.279428 11.495327  9.901481 11.471694  7.507988
## [22] 11.952988 11.256077 10.939270  8.544495 10.607596  8.555151  7.805335
## [29]  9.704344  8.605809 31.254490
```

The code for generating the boxplot of `sample` is as follows:

```
boxplot(sample,
        main="Boxplot of the sample which consists of 30 from N(10,2) & 1 from
              N(30,2)")
```



(b)

We can see that there is an extreme value (outlier) in the above figure , which is the element sampled from N(30,2).

(c)

Median and Inter-quartile range are the preferred choices to measure the location and spread of the distribution since they are not sensitive to extreme values whereas mean and standard deviation are. Since our distribution is skewed and has outlier as is shown in the boxplot, Median and the Inter-quartile range will be an appropriate measure in this case.

Below we compute each characteristic of `sample` and `sample[-31]` (outlier removed sample) to see its sensitivity to outliers. As shown below, the values of the mean and sd change substantially, while those of the median and IQR do not.

```
print(c(mean(sample),mean(sample[-31])))
```

```
## [1] 10.545162  9.854851
```

```
print(c(median(sample),median(sample[-31])))
```

```
## [1] 10.02185 10.00131
```

```
print(c(sd(sample),sd(sample[-31])))
```

```
## [1] 4.206371 1.738356
```

```
print(c(IQR(sample),IQR(sample[-31])))
```

```
## [1] 2.633218 2.544271
```

7. Evans & Rosenthal: 6.2.17

We first define the likelihood function `L` to compute likelihood by the following code:

```
L<-function(theta){
    return(exp(-(theta-1)^2/2)+3*exp(-(theta-2)^2/2))
}
```

Next, We generate 1000 equi-spaced points between -10 and 10 as follows:

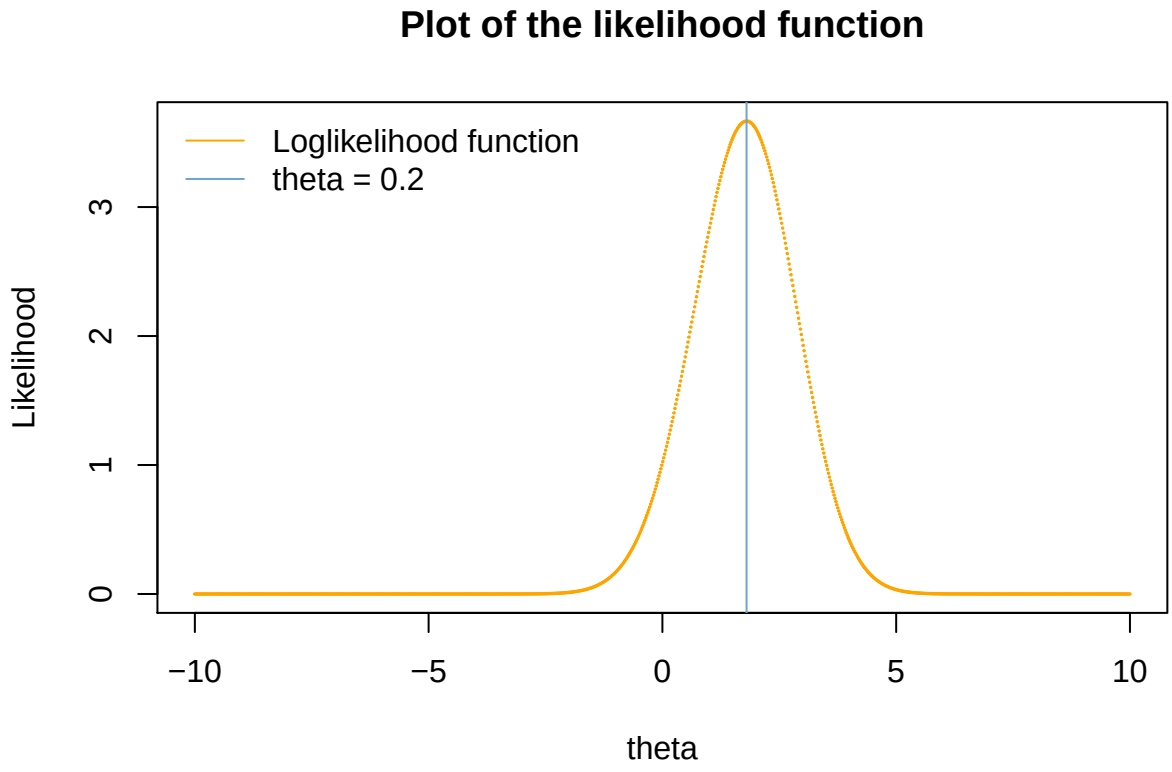
```
theta<-seq(-10, 10, length=1000)
```

We can numerically approximate MLE of $\theta = 1.87$. This can be generated by the following code:

```r
theta[which(L(theta)==max(L(theta)))]
```

```
## [1] 1.811812
```

```r
#Plot the likelihood function, L
plot(theta, L(theta), main = "Plot of the likelihood function", xlab="theta",
     ylab="Likelihood",cex=0.1, col= "orange")
abline(v = 1.8, col="skyblue3")
legend("topleft",c("Loglikelihood function","theta = 0.2"),
       col=c("orange","skyblue3")  , lty= c(1,1),bty='n')
```



8. Evans & Rosenthal: 6.2.25

(a)

Since n is assumed to be small relative to the size of the population, we can assume the observations are i.i.d.
We define random variables X_i ($i = 1, ..., 20$) that take 1 if ith individual is left-handed and 0 otherwise. We define $Y \equiv \sum_{i=1}^{20} X_i$. Since X_i is assumed to follow i.i.d. Bernoulli(θ), $Y \sim$ Binomial(20,θ). Therefore, the likelihood function $L(\theta|y)$ and log-likelihood function $l(\theta|y)$ are obtained as

$$L(\theta|y) = \theta^y(1-\theta)^{20-y},$$
$$l(\theta|y) = y\ln\theta + (20-y)\ln(1-\theta).$$

Since we observed 4 left-handed individuals, we substitute $y = 4$ into $l(\theta|y)$.
To obtain the MLE of θ, we have to solve $\partial l(\theta|y = 4)/\partial\theta = 0$.

$$\frac{\partial l}{\partial \theta}(\theta|y = 4) = \frac{4}{\theta} - \frac{16}{1-\theta} = \frac{4 - 20\theta}{\theta(1-\theta)}$$

Solving $\partial l(\theta|y = 4)/\partial\theta = 0$ gives the unique solution $\hat{\theta} = 0.2$.
To check that this is a local maximum, we calculate

$$\frac{\partial^2 l}{\partial \theta^2}(\theta|y = 4) = -\frac{4}{\theta^2} - \frac{16}{(1-\theta)^2}$$

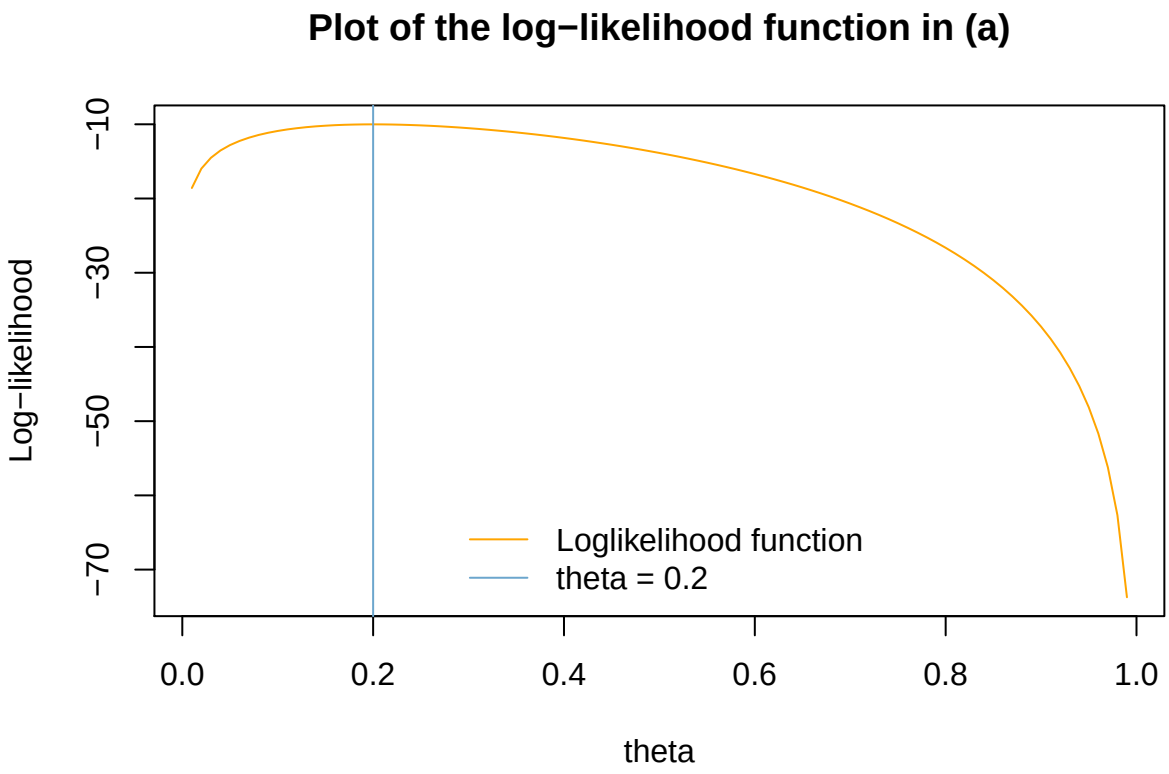
which is negative, and indicates that $\hat{\theta}$ is a local maximum. Since we have only one local maximum, it is also the global maximum and we have indeed obtained the MLE of θ as $\hat{\theta} = 0.2$.
To plot the log-likelihood function $l(\theta|y = 4)$, we define Loglikelihood function and substitute 100 equally spaced values in $(0, 1)$ into `theta` as follows:

```
#Define myloglikelihood
myloglikelihood <- function(theta){
  L <- 4*log(theta)+16*log(1-theta)
  return(L)
}

#Generate theta
theta <- seq(from=0,to=1,length.out=100+1)
theta <- theta[c(-1,-101)]

#Substitute theta into myloglikelihood
L <- myloglikelihood(theta)

#Plot the loglikelihood
plot(theta,L, main = "Plot of the log-likelihood function in (a)", xlab="theta",
     ylab="Log-likelihood", cex=0.5, type = "l", col="orange")
abline(v = 0.2, col="skyblue3")
legend("bottom",c("Loglikelihood function","theta = 0.2"),
       col=c("orange","skyblue3") , lty= c(1,1),bty='n')
```



We can see that $l(\theta|y)$ maximizes around $\theta = 0.2$ in the Figure. We can also numerically approximate the MLE of θ by the following R code:

```
theta_MLE <- theta[L==max(L)]
print(theta_MLE)
```

```
## [1] 0.2
```

(b)

Since the population size is 50, the population size of left-handed individuals is given by 50θ. Since we observe 4 left-handed individuals and 16 non-left-handed individuals, it holds that $50\theta \geq 4$ and $50(1-\theta) \leq 16$. In addition, 50θ must be an integer. Hence, the possible values of θ are given by $i/50$ where $i = 4, 5, ..., 34$. This is because the left-handed people can be atleast 4 and at most 34. Since the number of left-handed individuals follows Hypergeometric$(50, 50\theta, 20)$ and we observe 4 left-handed people, the likelihood function $L(\theta)$ and log-likelihood function $l(\theta)$ are obtained as

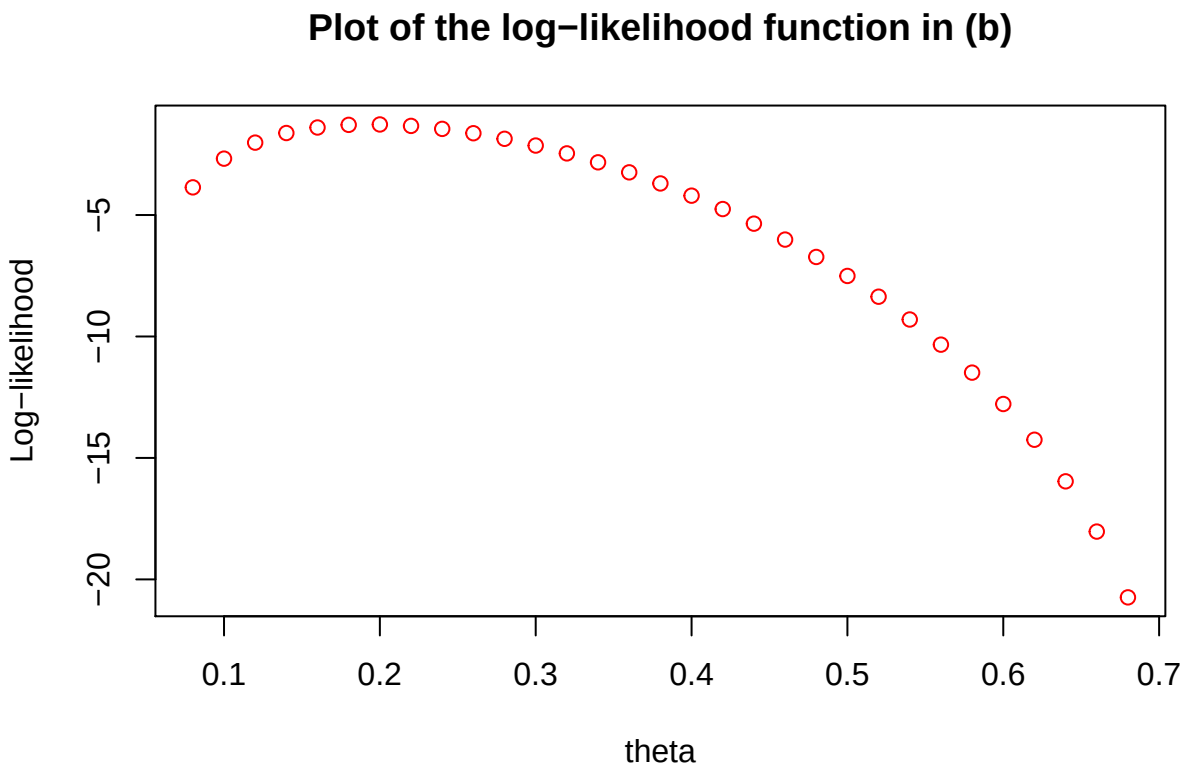
$$L(\theta) = \frac{\binom{50\theta}{4}\binom{50(1-\theta)}{16}}{\binom{50}{20}},$$
$$l(\theta) = \ln \frac{\binom{50\theta}{4}\binom{50(1-\theta)}{16}}{\binom{50}{20}}.$$

where $\theta = i/50$ $(i = 4, 5, ..., 34)$.
Hence, we can plot the log-likelihood function and determine the MLE by the following R code:

```r
#Define myloglikelihood_new
myloglikelihood_new <- function(theta){
 L <- log(dhyper(4, 50*theta, 50-50*theta, 20))
 return(L)
}

#Generate theta
theta <- (4:34)/50
L <- myloglikelihood_new(theta)

#Plot the loglikelihood
plot(theta,L, main = "Plot of the log-likelihood function in (b)", xlab="theta",
     ylab="Log-likelihood", col="red")
```




```r
#Get the MLE of theta
theta_MLE <- theta[L==max(L)]
print(theta_MLE)
```

```
## [1] 0.2
```

As is shown above, we have obtained the MLE of θ as $\hat{\theta} = 0.2$.

9. Evans & Rosenthal: 6.3.22

The following code generates 10^4 samples of size n =5 $\sim$ N(μ,σ^2) (where $\mu = 0$ and $\sigma^2 = 1$) and checks if $\mu \in (\overline{X} - S/\sqrt{n}, \overline{X} + S/\sqrt{n})$. It returns the total number of times the interval contains μ which gets stored in `count5`.

```
i=1
count5=0
for(i in 1:10^4){
  sample<-rnorm(5, mean = 0, sd = 1)
  if(mean(sample)-(sd(sample)/sqrt(5))<=0&&mean(sample)+(sd(sample)/sqrt(5))>=0){
    count5=count5+1
  }
  else{count5=count5}
}
```

We now compute the proportion of times the given interval contains μ by the following R code:

```
proportion5=count5/(10^4)
proportion5
```

```
## [1] 0.6291
```

We repeat this procedure with a larger n ($n = 10$ and then $n = 100$)

```
#n=10
count10=0
for(i in 1:10^4){
  sample<-rnorm(10, mean = 0, sd = 1)
  if(mean(sample)-(sd(sample)/sqrt(10))<=0&&mean(sample)+(sd(sample)/sqrt(10))>=0){
    count10=count10+1
  }
  else{count10=count10}
}
proportion10=count10/(10^4)
proportion10
```

```
## [1] 0.6623
```

```
#n=100
count100=0
for(i in 1:10^4){
  sample<-rnorm(100, mean = 0, sd = 1)
  if(mean(sample)-(sd(sample)/sqrt(100))<=0&&mean(sample)+(sd(sample)/sqrt(100))>=0){
    count100=count100+1
  }
  else{count100=count100}
}
proportion100=count100/(10^4)
proportion100
```

```
## [1] 0.6821
```

We observe the following: When $n = 5$, the proportion of μ in the interval is 0.6261. When $n = 10$, the proportion of μ in the interval is 0.6527. When $n = 100$, the proportion of μ in the interval is 0.6758. The proportion gets bigger when sample size is larger.

Also, as n grows, the given interval gets narrower

10. Evans & Rosenthal: 6.4.17

First we obtain the Maximum Likelihood Errors (MLEs) of μ and σ^2. Since the log-likelihood function of N(μ,σ^2) is given by

$$l(\mu, \sigma^2 | x_1, ..., x_n) = -\frac{n}{2} \ln 2\pi - \frac{n}{2} \ln \sigma^2 - \frac{n}{2\sigma^2}(\overline{x} - \mu)^2 - \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{2\sigma^2}$$

where x_i $(i = 1, ..., n)$ are i.i.d. samples from N(μ,σ^2) and $\overline{x} \equiv \sum_{i=1}^{n} x_i/n$. By solving $\partial l/\partial \mu = \partial l/\partial \sigma^2 = 0$, (Basically the first order condition) we obtain the MLEs of μ and σ^2 as follows:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i, \hat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

Hence, we can compute the plug-in MLE of $F(3)$, which we call $\hat{F(3)}$, as `F3hat` in the following R code:

```
data<- c(3.27, -1.24, 3.97, 2.25, 3.47, -0.09, 7.45, 6.20, 3.74, 4.12, 1.42,
         2.75, -1.48, 4.97, 8.00, 3.26, 0.15, -3.64, 4.88, 4.55)
muhat <- mean(data)
sigma_sqhat <- mean((data-muhat)^2)
F3hat <- pnorm(3, mean=muhat, sd=sqrt(sigma_sqhat))
print(F3hat)
```

```
## [1] 0.5136531
```

Now, we define a function that generates m bootstrap samples of $\hat{F(3)}$ as follows:

```
mybootF3hat <- function(data,m){
  #Generate m bootstrap samples
  boot_samples <- NULL
  for(i in 1:m){
    sample_i <- sample(data,size=length(data),replace=TRUE)
    boot_samples <- rbind(boot_samples,sample_i)
  }
  dimnames(boot_samples) <- NULL

  #Obtain samples of mu_hat and sigma^2_hat by using bootstrapping
  boot_muhat <- numeric(m)
  boot_sigma_sqhat <- numeric(m)
  for(i in 1:m){
    boot_muhat[i] <- mean(boot_samples[i,])
    boot_sigma_sqhat[i] <- mean((boot_samples[i,]-boot_muhat[i])^2)
  }

  #Obtain samples of F(3)_hat by plugging-in mu_hat and sigma^2_hat
  boot_F3hat <- pnorm(3, mean=boot_muhat, sd=sqrt(boot_sigma_sqhat))
```

```
  return(boot_F3hat)
}
```

Since the MSE of $\hat{F(3)}$ is obtained by the following:

$$\begin{aligned} MSE_{\mu,\sigma^2}(F(3)) &= E_{\mu,\sigma^2}((\hat{F(3)} - F(3))^2) \\ &= (E_{\mu,\sigma^2}(\hat{F(3)}) - F(3))^2 + Var_{\mu,\sigma^2}(\hat{F(3)}) \\ &= Bias_{\mu,\sigma^2}(\hat{F(3)})^2 + Var_{\mu,\sigma^2}(\hat{F(3)}), \end{aligned}$$

We can estimate $MSE_{\mu,\sigma^2}(F(3))$ by bootstrapping in two ways: (i) either estimate $MSE_{\mu,\sigma^2}(F(3))$ directly or (ii) estimate $Bias_{\mu,\sigma^2}(\hat{F(3)})^2$ and $Var_{\mu,\sigma^2}(\hat{F(3)})$ then sum them up. By bootstrapping for $m = 10^3$, we can estimate $MSE_{\mu,\sigma^2}(F(3))$, $Bias_{\mu,\sigma^2}(\hat{F(3)})^2$ and $Var_{\mu,\sigma^2}(\hat{F(3)})$ as follows:

```
#Generate bootstrap samples
set.seed(430)
boot_F3hat_3 <- mybootF3hat(data,10^3)

#(i) Estimate MSE directly
MSEhat_3_i <- mean((boot_F3hat_3-F3hat)^2)

#or

#(ii) Estimate Bias^2 and Var then sum them up
boot_bias_3 <- mean(boot_F3hat_3)-F3hat
boot_var_3 <- var(boot_F3hat_3)
MSEhat_3_ii <- boot_bias_3^2+var(boot_F3hat_3)

#Show the results:
print(c(MSEhat_3_i,MSEhat_3_ii))
```

```
## [1] 0.009310247 0.009319538
```

We can simulate for $m = 10^4$ in the same way:

```
#Generate bootstrap samples
set.seed(430)
boot_F3hat_4 <- mybootF3hat(data,10^4)

#Estimate MSE directly
MSEhat_4_i <- mean((boot_F3hat_4-F3hat)^2)

#Estimate Bias^2 and Var then sum them up
boot_bias_4 <- mean(boot_F3hat_4)-F3hat
boot_var_4 <- var(boot_F3hat_4)
MSEhat_4_ii <- boot_bias_4^2+var(boot_F3hat_4)

#Show the results
print(c(MSEhat_4_i,MSEhat_4_ii))
```

```
## [1] 0.008899025 0.008899911
```

We can see that the two estimates of the MSE of $F(\hat{3})$ are close enough for both $m = 10^3$ and 10^4.
Summarizing above, the estimate of the MSE of $F(\hat{3})$ is about 0.00931 for $m = 10^3$ and about 0.00890 for $m = 10^4$.