Homework 1

Yuanjian Zhou

Last modified Oct 9

Contents

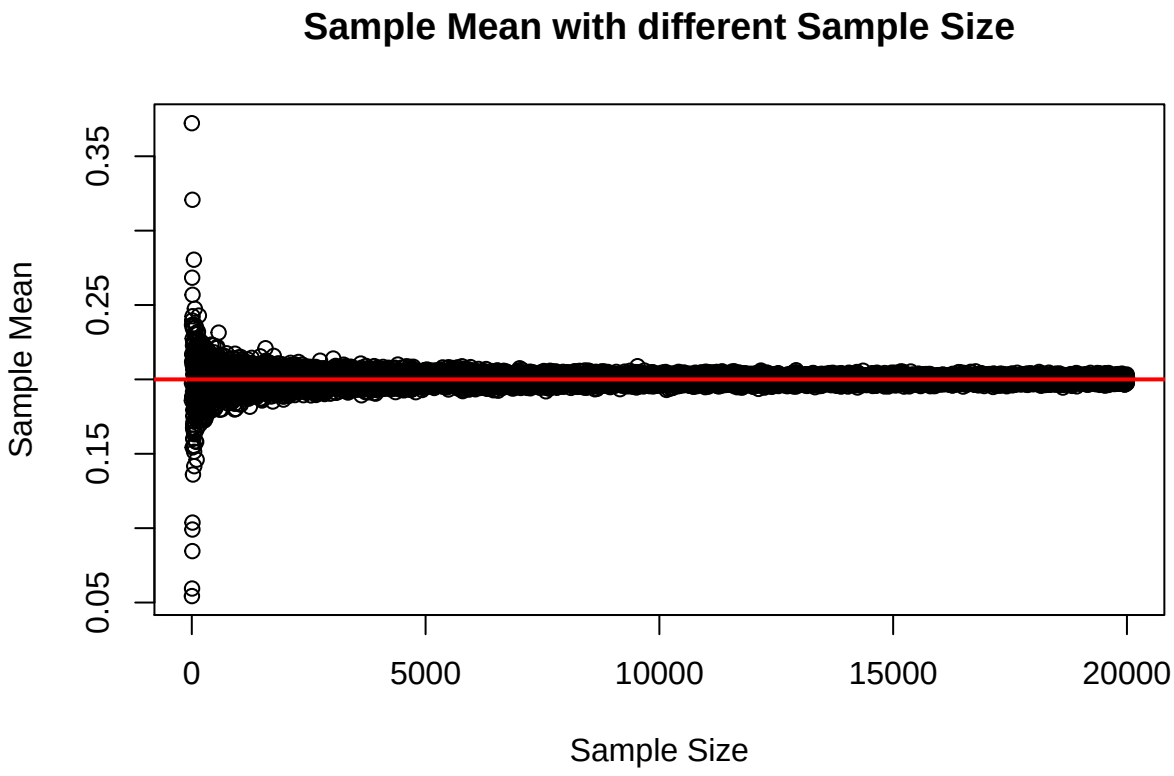
Question 1

First I define the function to generate the sample mean.

```
set.seed(46) # make the results replicable
Exponential_sample_mean = function(n){
  X = rexp(n, 5)
  return (mean(X))
}
```

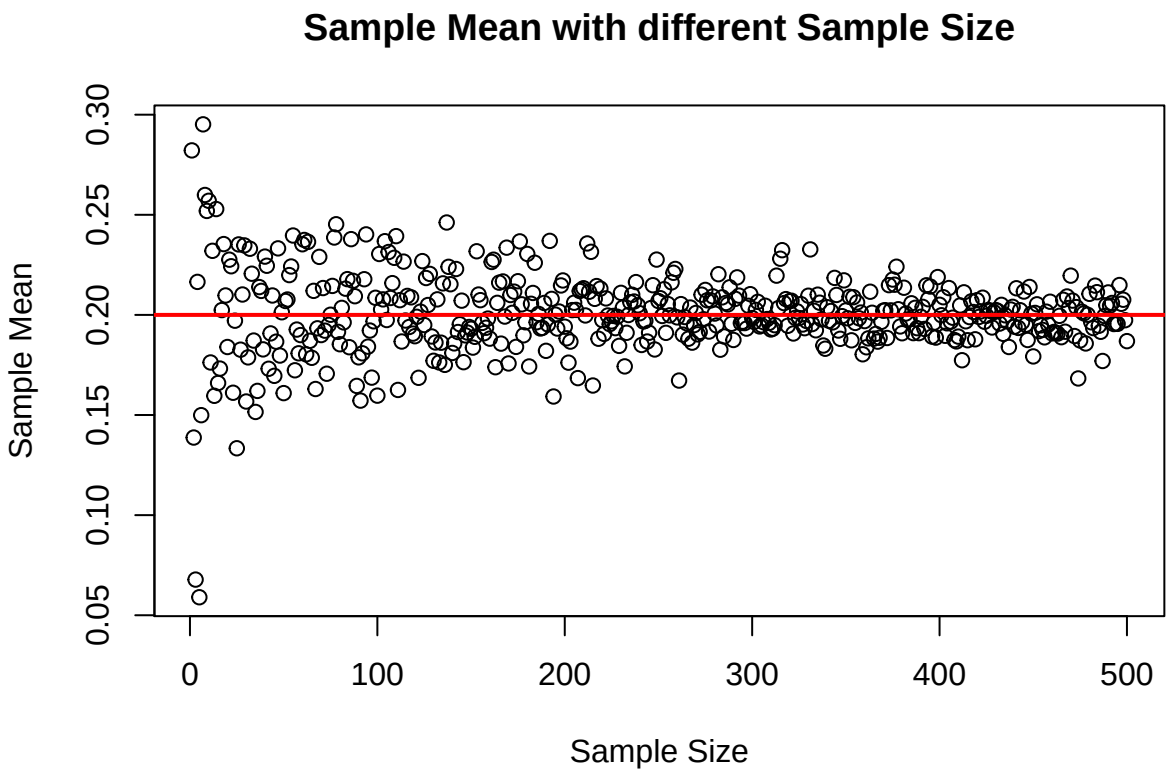
Then I calculate the sample mean $M_1, M_2, ..., M_n$, and plot them.

```
n = c(1 : 20000)# 10^5 takes too much time to run
Mn = unlist(lapply(n, Exponential_sample_mean)) #here lapply returns a list and we can not use a list t
plot(Mn, xlab = "Sample Size", main = "Sample Mean with different Sample Size", ylab = "Sample Mean")
abline(h = 0.2, col = "red", lwd = 2)
```



From the graph I find that they are converging to 0.2 very quickly. I can draw a new graph with sample from 1 to 500.

```
n = c(1 : 500)
Mn = unlist(lapply(n,Exponential_sample_mean))
plot(Mn, xlab = "Sample Size", main = "Sample Mean with different Sample Size", ylab = "Sample Mean")
abline(h = 0.2, col = "red", lwd = 2)
```



From the graph I can see that it starts toconverge to 0.2 when the sample size is about 300.

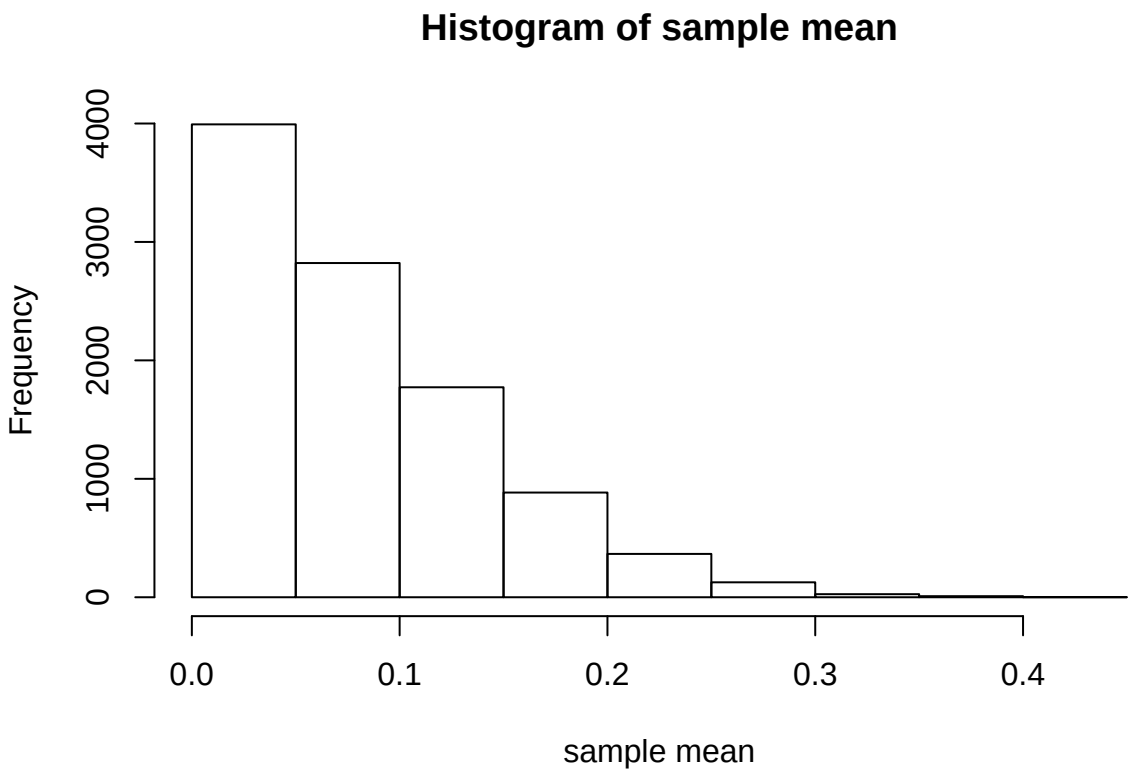
Question 2

First I define the function to generate the sample mean.

```
set.seed(46)
Binom_sample_mean = function(n = 20, p = 0.01){
  X = rbinom(n, 10, p)
  return (mean(X))
}
```

Then I replicate this function for 10^4 times and construct the density histogram of the sample mean.

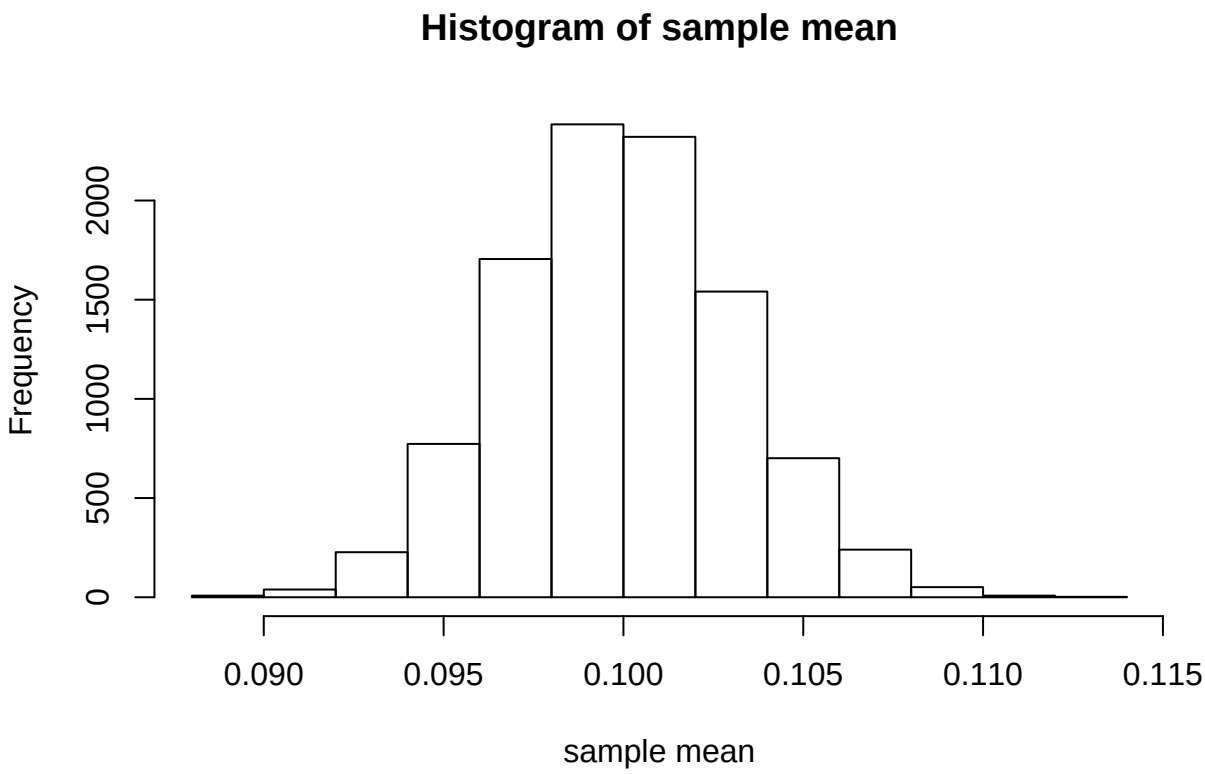
```
sample_mean_10e4 = replicate(10000, Binom_sample_mean(20, 0.01))
hist(sample_mean_10e4, main ="Histogram of sample mean", xlab = "sample mean")
```



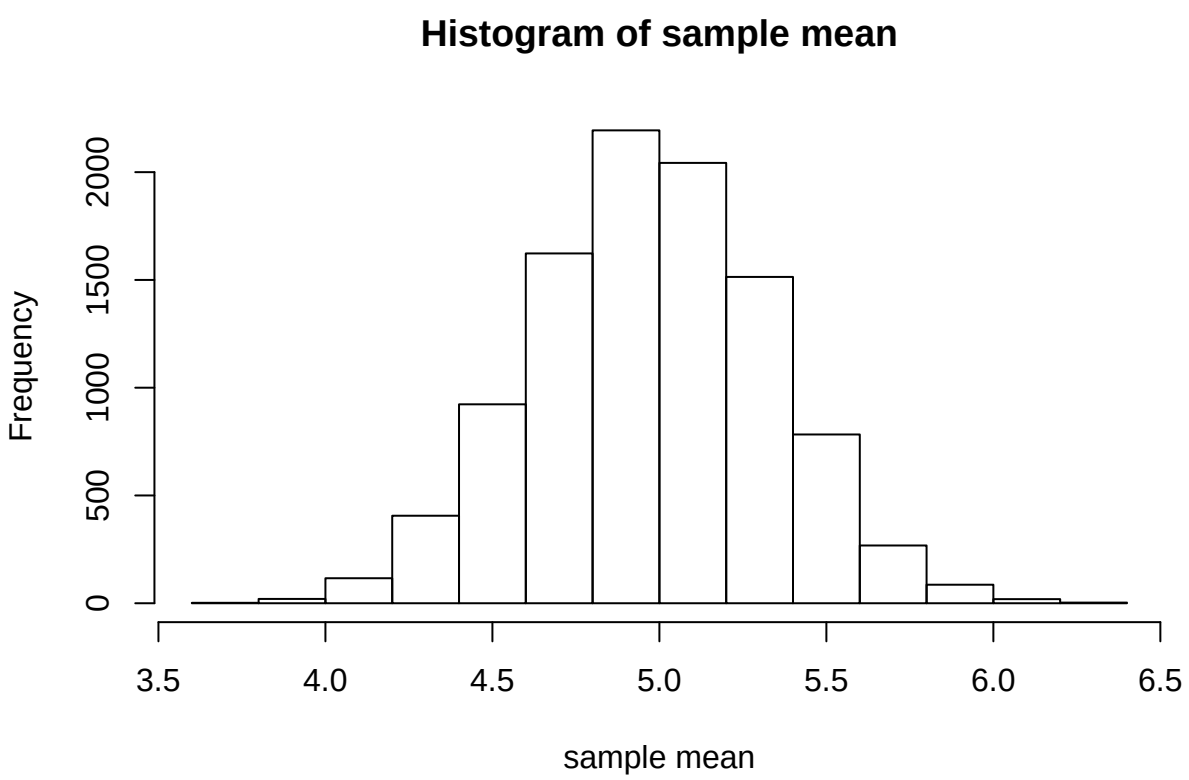
I think that the shape of this graph does not look like a normal distribution. But it does not mean Central Limit Theorem does not work, I think it is because the sample size is too small or the probability is too small.

Then I test these two reasons.

```
sample_mean_10e4 = replicate(10000, Binom_sample_mean(10000,0.01))
hist(sample_mean_10e4, main ="Histogram of sample mean", xlab = "sample mean")
```




```
sample_mean_10e4 = replicate(10000, Binom_sample_mean(20,0.5))
hist(sample_mean_10e4, main ="Histogram of sample mean", xlab = "sample mean")
```



I can find that either you increase the probability or increase the sample size it will look more like a Normal distribution.

Question 3

a)

I first calculate the probability:

$$P(Y \leq c_1|p = 0.4), P(Y \geq c_2|p = 0.4)$$

for each possible c_1 and c_2. In order to see clearly, I make a data frame.

```
c1 = c(0:9)
c2 = c(0:9)
probability_y_lessthanorequalc1 = pbinom(c1, 9, 0.4)
probability_y_morethanorequalc2 = 1-pbinom(c2-1, 9, 0.4)
df = data.frame(probability_y_lessthanorequalc1,probability_y_morethanorequalc2, row.names = c(0:9))
df
```

```
##   probability_y_lessthanorequalc1 probability_y_morethanorequalc2
## 0                      0.01007770                     1.000000000
## 1                      0.07054387                     0.989922304
## 2                      0.23178701                     0.929456128
```

```
## 3                0.48260966              0.768212992
## 4                0.73343232              0.517390336
## 5                0.90064742              0.266567680
## 6                0.97496525              0.099352576
## 7                0.99619891              0.025034752
## 8                0.99973786              0.003801088
## 9                1.00000000              0.000262144
```

From the frame I can see that if $c_2 = 0.6$, then the probability $P(Y \geq c_2|p = 0.4)$ is really close to 0.1. But if $c_2 = 0$, the sum of the probability will go over 0.1. And I know that Y can not be a negative number, so if I let $c_1 < 0$,then the probability $P(Y \leq c_1|p = 0.4)$ will be 0. Then the sum of the two probabilities is the nearest to 0.1 without exceeding 0.1. So $c_1 < 0$, for example $c_1 = -1$ and $c_2 = 6$.

b)

According to the definition of the size of the test, it is just the probability that we reject the null hypothesis. So the size of the test δ is just

$$P(Y \leq -1|p = 0.4), P(Y \geq 6|p = 0.4) = 0.0994$$

c)

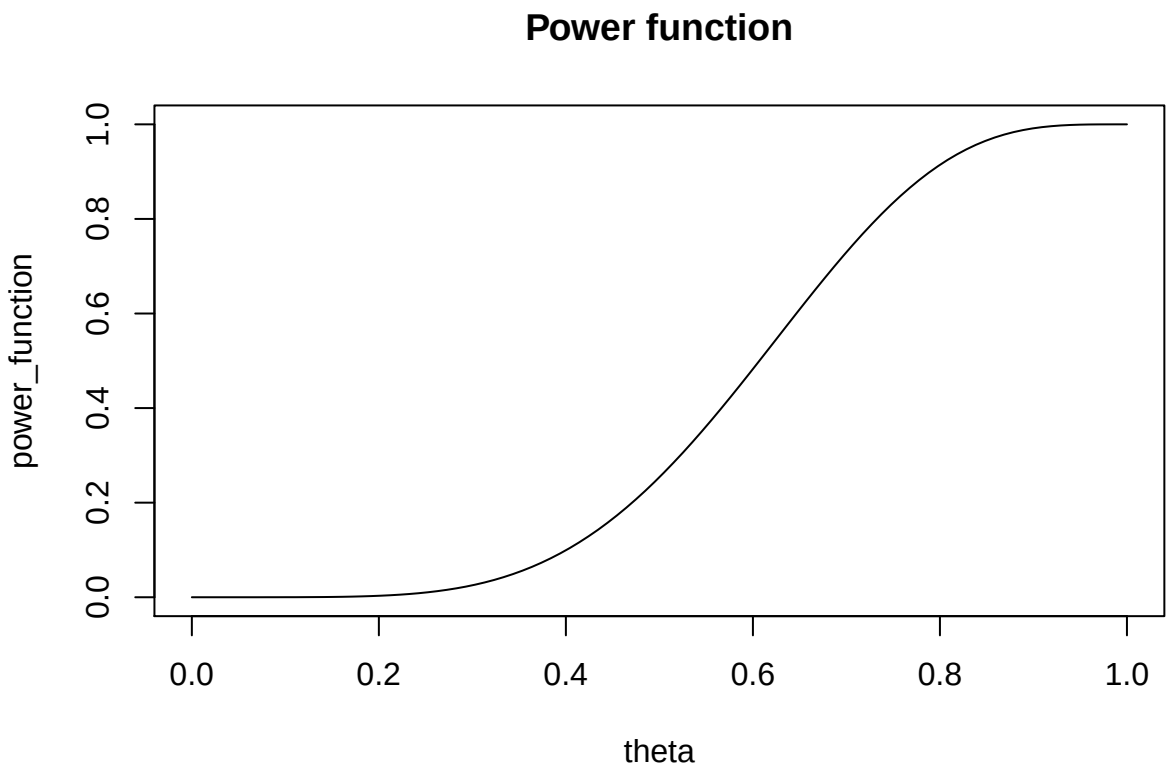
According to the definition of the power function, which is

$$\pi(\theta|\delta) = Pr(X \in S_1|\theta) \ fortheta \in \Omega$$

Here S_1 denotes the reject region.

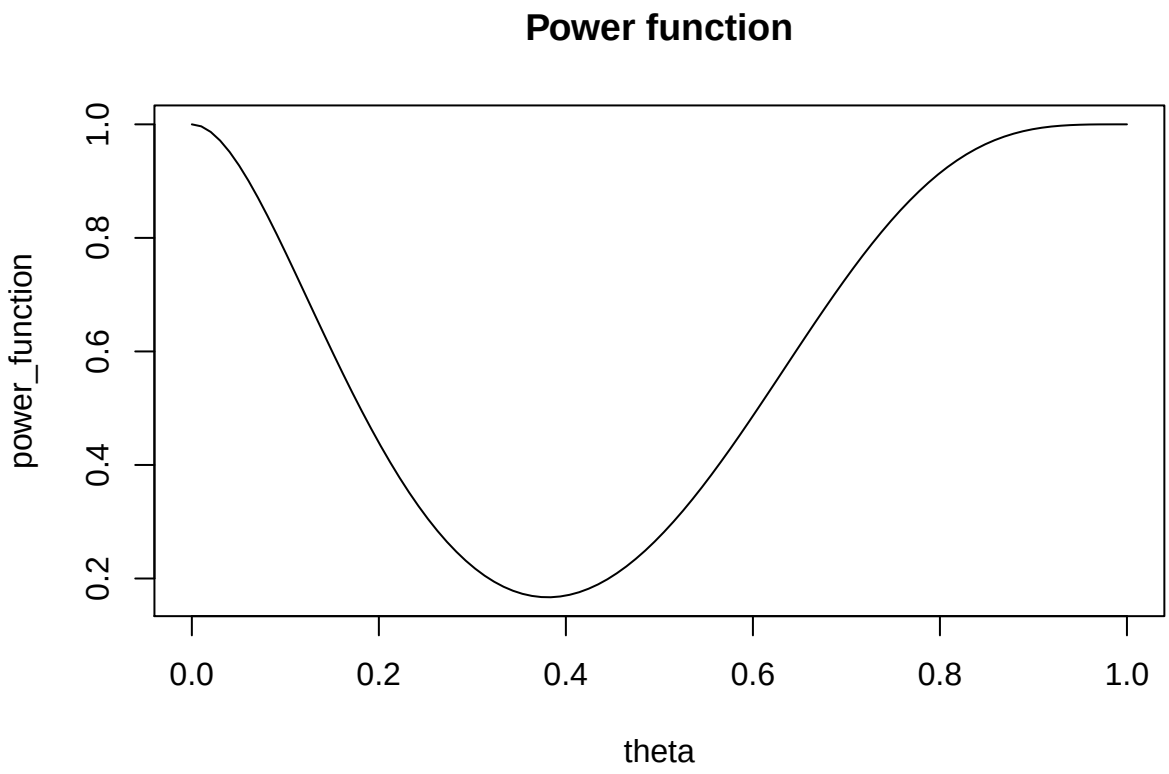
Then I can get the power function for this test.

```
theta = seq(0, 1, length = 100)
power_function = 1 - pbinom(6 - 1, 9, theta)
plot(theta,type = "l" , power_function, xlab = "theta", main = "Power function")
```



As we can see from the graph, this test works bad when $p < 0.4$, and it is because since $c_1 < 0$, so the size of the test is just $P(Y \geq 6)$. Instead we can use $c_1 = 1$,$c_2 = 7$, although the size is a litter bit larger, it works better when $p < 0.4$.

```
power_function = 1 - pbinom(6 - 1, 9, theta) + pbinom(1, 9, theta)
plot(theta, type = "l", power_function, xlab = "theta", main = "Power function")
```

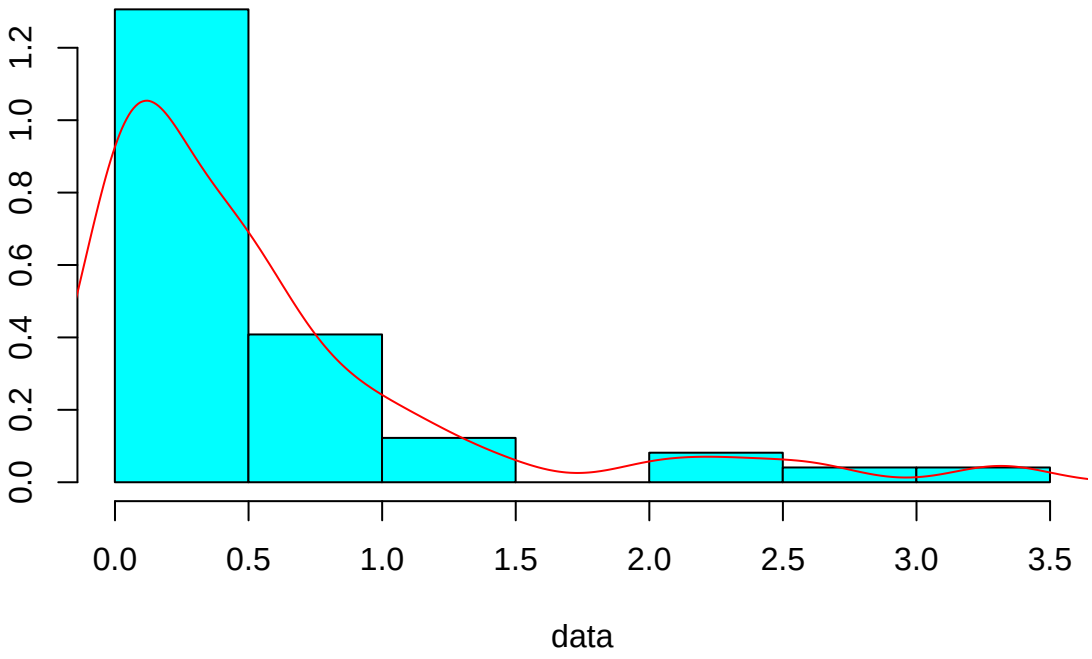


Question 4

a)

First I import the data and draw histgram and density curve.

```
csv = read.csv("Prob4_data.txt")
data = csv[[1]] # We need a numeric series instead of a dataframe.
library(MASS)
truehist(data) # truehist function works better if you want to draw the density curve.
lines(density(data),col = "red")
```



b)

From the note of our professor, I know that the moment of estimate of the parameters of gamma distribution is

$$\hat{\alpha} = \frac{\bar{X}^2}{\frac{1}{n}\sum_{i=1}^{n} X_i^2 - \bar{X}^2} \hat{\beta} = \frac{\bar{X}}{\frac{1}{n}\sum_{i=1}^{n} X_i^2 - \bar{X}^2}$$

So I just need to calculate them with my data.

```
a_mom = mean(data)^2 / (mean(data^2) - mean(data)^2)
b_mom = mean(data) / (mean(data^2) - mean(data)^2)
```

c)

First I need three variables to store my results of bootstrap.

```
bootstrap_mean = rep(0,1000)
bootstrap_a = rep(0,1000)
bootstrap_b = rep(0,1000)
```

Then I do the bootstrap and get these statistics.

```
for (i in 1:1000) {
  bootstrap_sample = sample(data,length(data),replace = T)
  bootstrap_mean[i] = mean(bootstrap_sample)
  bootstrap_a[i] = (bootstrap_mean[i]^2) / (mean(bootstrap_sample^2) - bootstrap_mean[i]^2)
  bootstrap_b[i] = (bootstrap_mean[i])/ (mean(bootstrap_sample^2) - bootstrap_mean[i]^2)
}
```

Afterwards accroding to formula in the notes:

$$\text{Mean:}\bar{\hat{\psi}} = \frac{1}{n}\sum_{i=1}^{m}\hat{\psi}_i \text{Standard Error:} se(\hat{\psi}) = \sqrt{\widehat{Var}_{\hat{F}}(\hat{\psi})}$$

Here ψ is the statistic that I am interested in.

Then I can calculate the Bootstrap_mean, Bootstrap_a and Bootstrap_b and the standard error of them.

```
standard_error = sd(bootstrap_mean)
Bootstrap_mean = mean(bootstrap_mean)
Bootstrap_a = mean(bootstrap_a)
Bootstrap_b = mean(bootstrap_b)
```

Then, according to the formula of confidence interval, which is:

$$\hat{\psi} \pm t_{(1+\gamma)/2,n-1} * se(\hat{\psi})$$

I can get the 95% confidence interval of a, b

```
left_ci_a = Bootstrap_a-pt(0.975,999)*sd(bootstrap_a)
right_ci_a = Bootstrap_a+pt(0.975,999)*sd(bootstrap_a)
left_ci_b = Bootstrap_b-pt(0.975,999)*sd(bootstrap_b)
right_ci_b = Bootstrap_b+pt(0.975,999)*sd(bootstrap_b)
cat("the 95% ci for a is","[",left_ci_a, right_ci_a,"]","\n","the 95% ci for b is","[",left_ci_b, right_
```

```
## the 95% ci for a is [ 0.5189997 0.775784 ]
##  the 95% ci for b is [ 0.9039643 1.543809 ]
```

Lastly, I compare them with b)

```
a_mom
```

```
## [1] 0.5968179
```

```
Bootstrap_a
```

```
## [1] 0.6473918
```

```
b_mom
```

```
## [1] 1.091482
```

```
Bootstrap_b
```

```
## [1] 1.223886
```

I find that there is just a little difference between them.

Question 5

First I make the population space.

```
population = seq(1, 10000)
set.seed(46)
```

Then I generate the sample as the question asks.

a)

```
simple_random_sample = sample(500, population)
```

b)

```
iid_sample = sample(500, population, replace = T)
```

Question 6

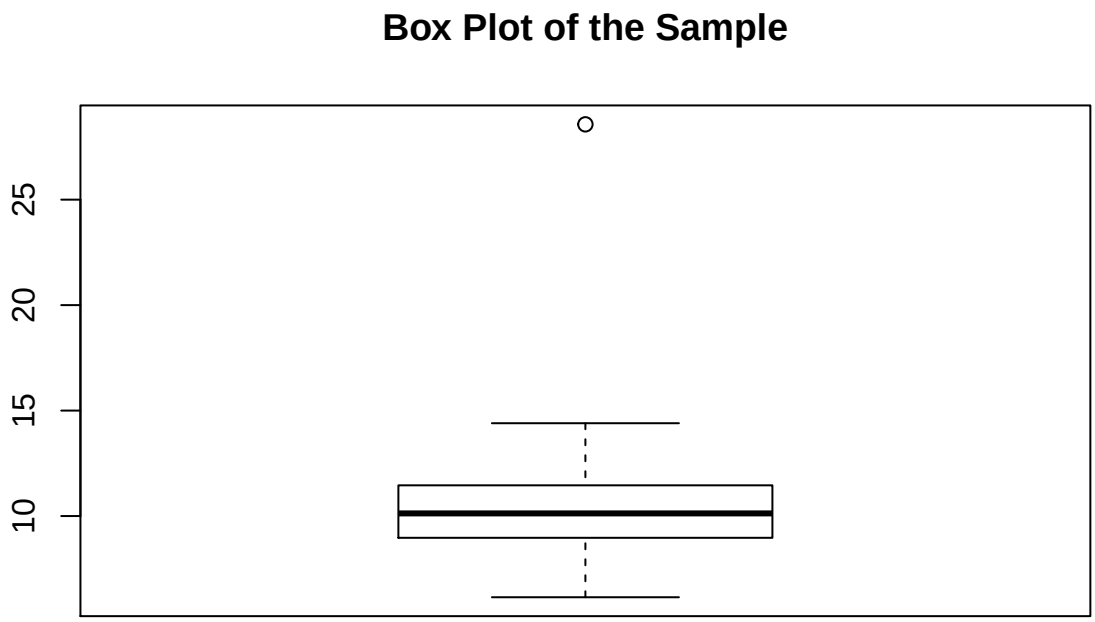
a)

First I generate these two samples.

```
set.seed(46)
sample1 = rnorm(30, 10, 2)
sample2 = rnorm(1, 30, 2)
```

Then I combine these samples and draw the boxplot.

```
sample = c(sample1, sample2)
boxplot(sample, main = "Box Plot of the Sample")
```



b)

There is a outlier in this plot, and the length of the box is short.

c)

Based on the boxplot, I think that mean, median, IQR, upper bound, lower bound, and outliers are appropriate for measuring the location, since with the mean, we can know that what the data is centered with , but the mean is sensitive to outliers, so we may also need median. With IQR, upper bound, and lower bound we can know how the data is spreaded, with outliers we can know whether there is a strange point away from the others.

Question 7

First I need a function to express the likelihood.

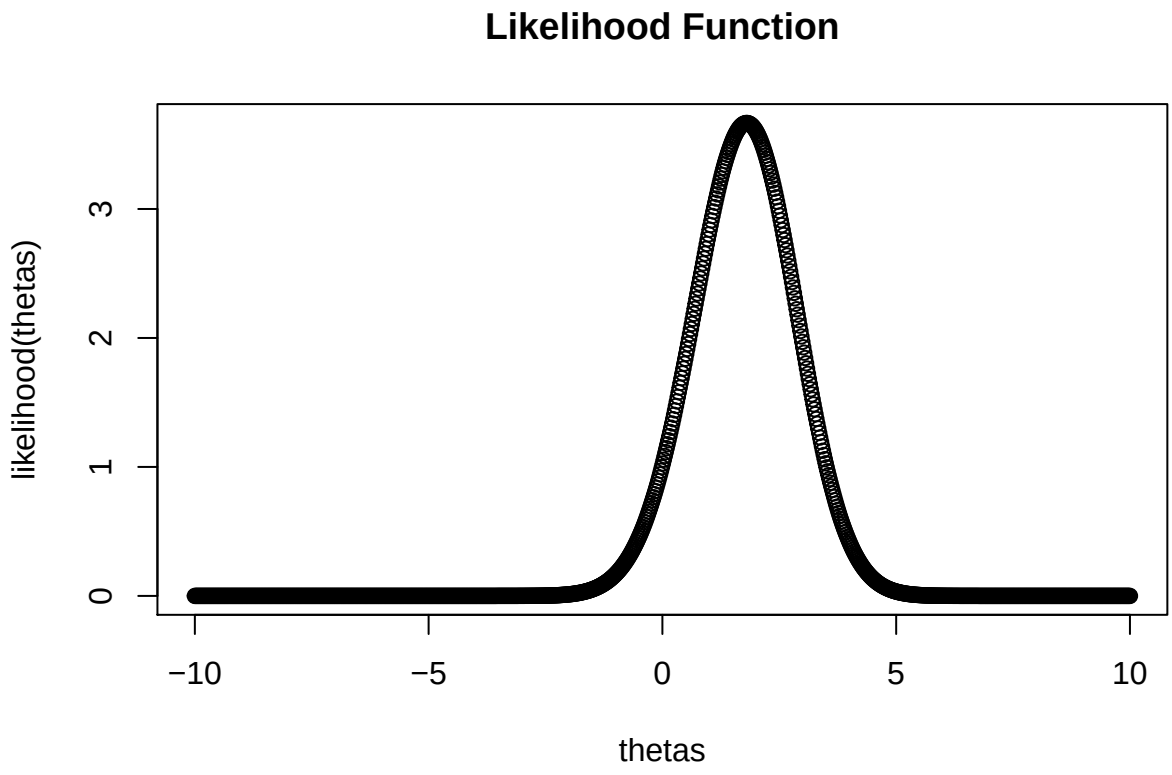
```
likelihood = function(theta){
  likelihood = exp((-(theta-1)^2)/2)+3*exp(-((theta-2)^2)/2)
  return (likelihood)
}
```

Then I generate the θ, approximate the MLE estimator and draw the graph.

```r
thetas= seq(-10, 10, length.out = 1000)
MLE = thetas[which.max(likelihood(thetas))]
MLE
```

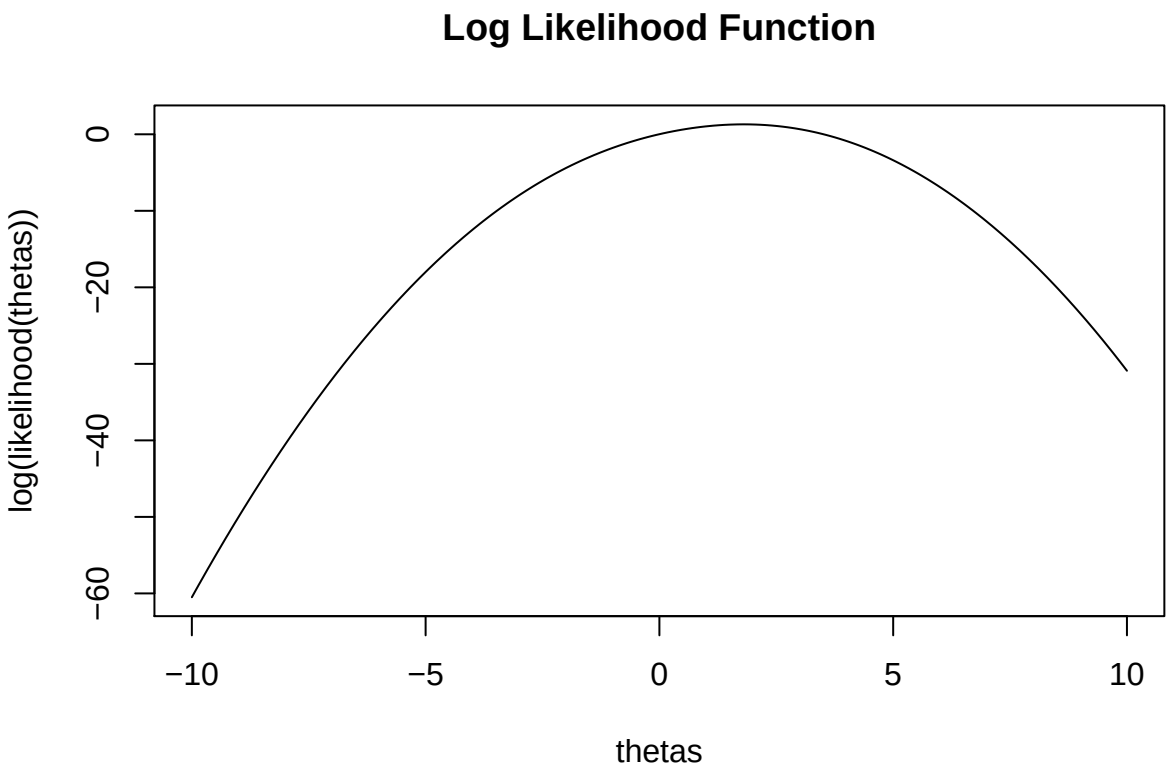
```
## [1] 1.811812
```

```r
plot(thetas, likelihood(thetas), main = "Likelihood Function")
```



Although the question asks I to draw the likelihood function, in reality, we usually use log likelihood.

```r
plot(thetas, type = "l", log(likelihood(thetas)), main = "Log Likelihood Function")
```



Question 8

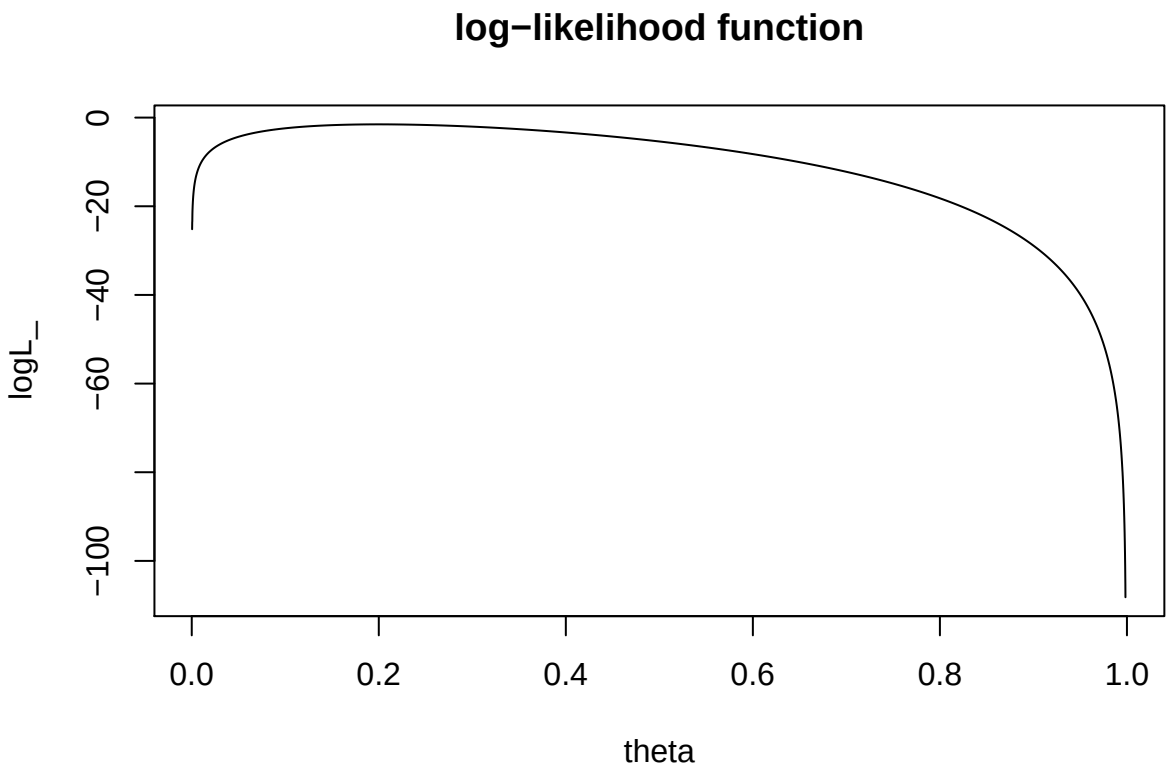
a)

First from the scene desribed in the text, I can see that the event that you observe left-handed individuals obeys hyper geometric distribution. So we can define the likelihood function.

```
L_lefthand = function(x = 4, N, M, k = 20){ # Here M is the left_handed in the polulation, N is the po
  return(choose(M,x)*choose(N-M, k-x)/choose(N,k))
}
```

Then I generate thetas, and calculate the likelihood for each thetas, afterwards I draw the log-likelihood function and get the MLE estimator.

```
thetas_ = c(1:10000)/10000
logL_ = log(L_lefthand(N = 10000, M = 10000 * thetas_ ))
plot(thetas_, type = "l", logL_, main = "log-likelihood function", xlab = "theta")
```



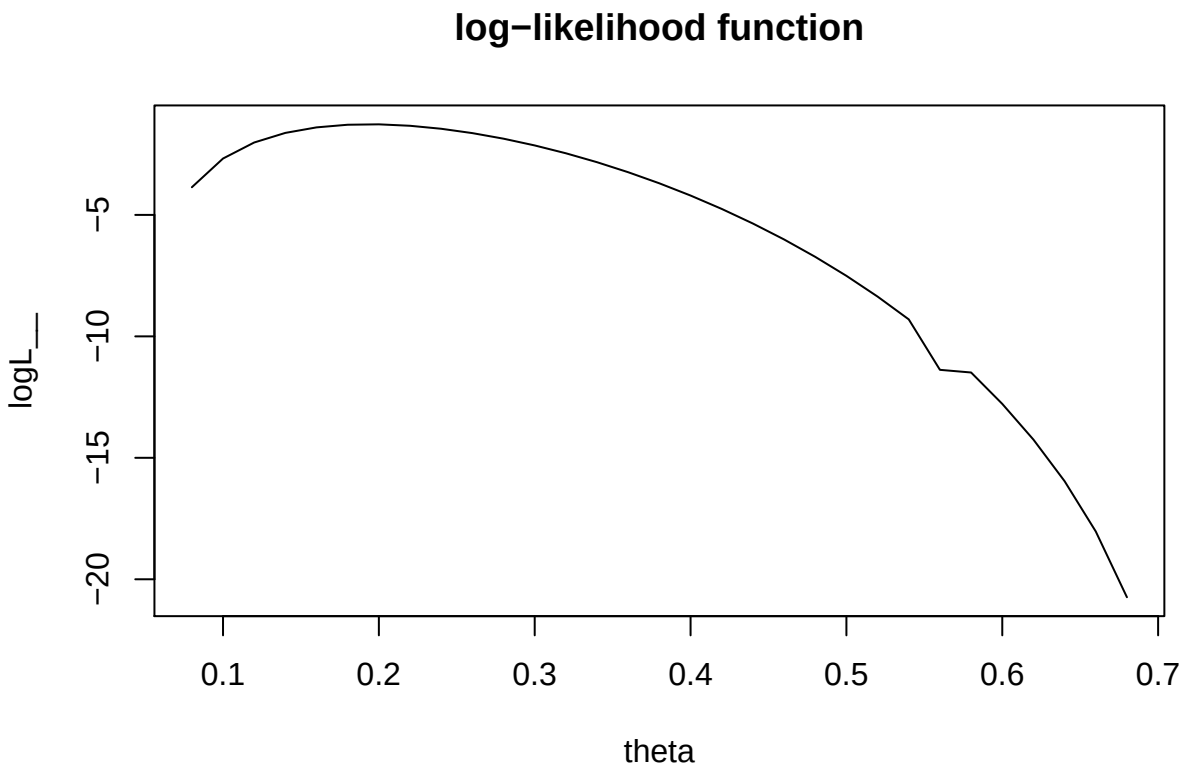

```
MLE_ = thetas_[which.max(logL_)]
MLE_
```

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

b)

Change a) a little bit, we can get the answer.

```
thetas__ = c(4:34)/50
logL__ = log(L_lefthand(N = 50, M = 50 * thetas__ ))
plot(thetas__, type = "l", logL__, main = "log-likelihood function", xlab = "theta")
```




```
MLE__ = thetas__[which.max(logL__)]
MLE__
```

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

Question 9

First I generate the sample and calculate whether μ is in the interval.

```
set.seed(46)
isin_interval = function(n){
  sample = rnorm(n, 0 ,1)
  Normal_mean = mean(sample)
  Normal_sd = sd(sample)
  if (Normal_mean + Normal_sd/sqrt(n) >0 && Normal_mean - Normal_sd/sqrt(n) < 0){
    return(1) # 1 means mu is in the interval, 0 means not
  }else{
    return(0)
     }
}
```

Then I replicate the function 10^5 times, and see how many times the function return 1.

```
Total_sample_5 = replicate(10000,isin_interval(5))
prop_5 = sum(Total_sample_5)/10000
prop_5
```

```
## [1] 0.6318
```

Similarly I can calculate the prop when n = 10 and 100.

```
Total_sample_10 = replicate(10000,isin_interval(10))
prop_10 = sum(Total_sample_10)/10000
prop_10
```

```
## [1] 0.6575
```

```
Total_sample_50 = replicate(10000,isin_interval(50))
prop_50 = sum(Total_sample_50)/10000
prop_50
```

```
## [1] 0.6774
```

Question 10

First I import the data.

```
set.seed(46)
X<-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,
```

Then I plug in the MLE to get the estimator of F_3, and calculate the true F_{3}.

```
F3_hat = pnorm((3 - mean(X))/sd(X))
F3 = length(X[which(X < 3)])/length(X)
bias = (F3_hat - F3)^2
```

Next I define the bootstrap function and get the estimator of F3 for each time.

```
bootstrap = function(){
  X_bootstrap = sample(X, length(X), replace = T)
  F3_hat_boot = pnorm((3 - mean(X_bootstrap))/sd(X_bootstrap))
  return(F3_hat_boot)
}
```

Then I do the bootstraps for 10^3 times and calculate MSE

```
bootstrap_1000 = replicate(1000,bootstrap())
bias_1000 = (mean(bootstrap_1000) - F3)^2
boostrap_1000_MSE = bias_1000 + var(bootstrap_1000)
boostrap_1000_MSE
```

```
## [1] 0.02126766
```

Similarly,I do it for 10^4 times.

```
bootstrap_10000 = replicate(10000,bootstrap())
bias_10000 = (mean(bootstrap_10000) - F3)^2
boostrap_10000_MSE = bias_10000 + var(bootstrap_10000)
boostrap_10000_MSE
```

```
## [1] 0.02036462
```