ECON 430 Homework 1

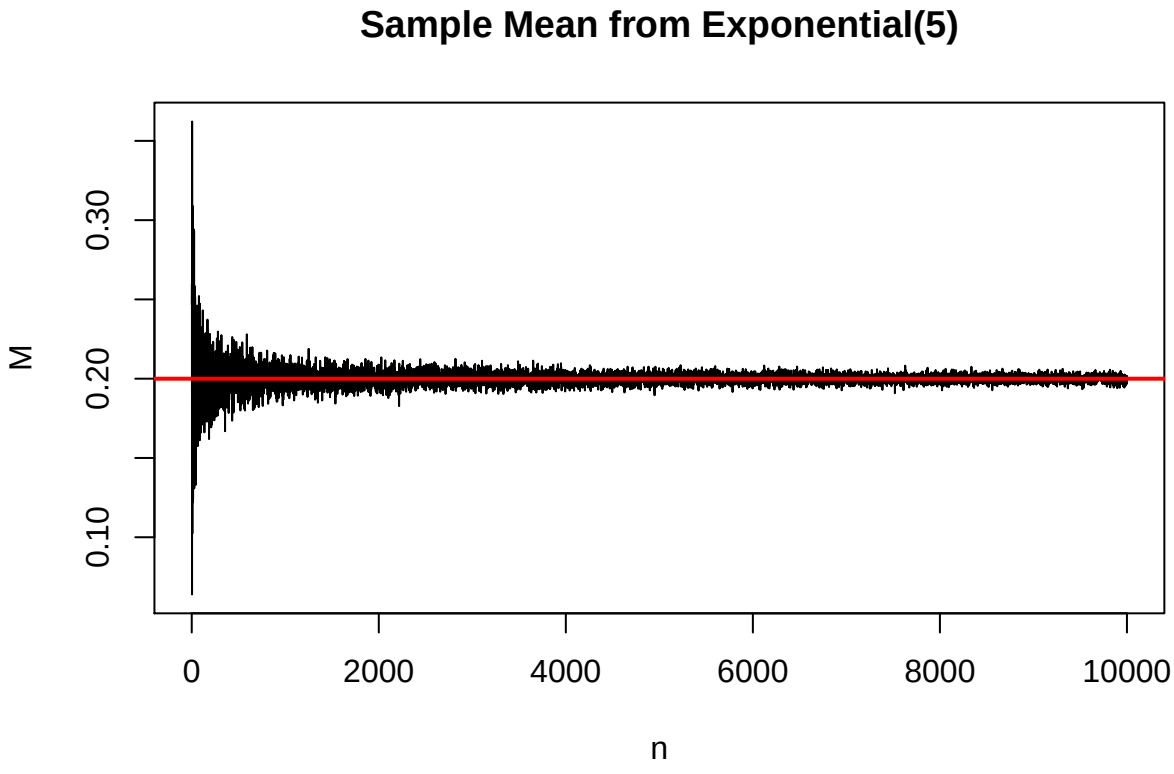
Gefei Zhao

2020/10/10

Exercise 4.3.13

```r
# generate Mn of exp(5)
Mn <- function(n){
  return(mean(rexp(n, rate=5)))
}
#take sample size = 10^4
M <- unlist(lapply((1:10^4), Mn))

plot(M, type='l', xlab="n", main="Sample Mean from Exponential(5)")
abline(h=0.2, col="red", lwd=2)
```

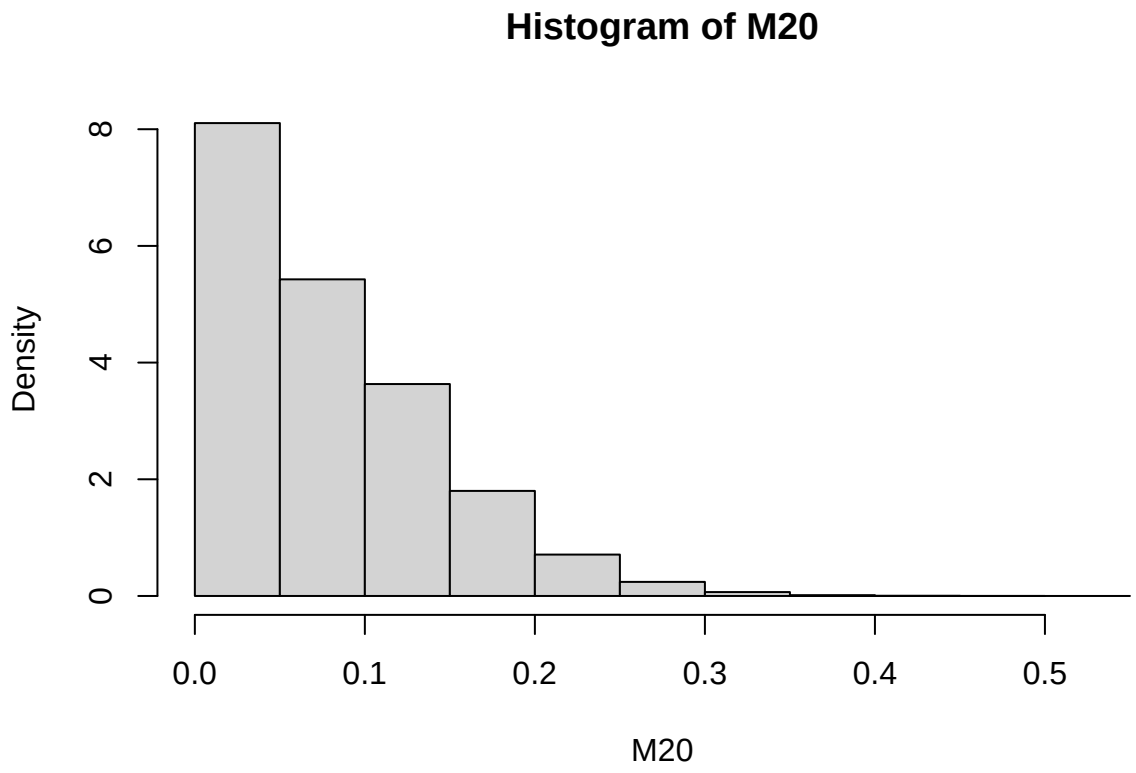


It is obvious in the plot that $M_1, M_2, ..., M_n$ are converging quickly to 0.2 which is the μ of $Exponential(5)$.

Exercise 4.4.19

```
set.seed(1234)
Mn <- function(n){
  x <- rbinom(n, 10, 0.01)
  return(mean(x))
}

M20 <- replicate(10^5, Mn(20))
hist(M20, prob=T)
```



We can found that the data of highest density is around 0.1 which is the mean of distribution.

Exercise 4.5.15

```
## Calculate the value of the integral directly
set.seed(1)
f_0 <- function(x){
  25*cos(x^4) * exp(-25*x)
}
mu <- integrate(f_0, 0, Inf)
print(mu$value)
```

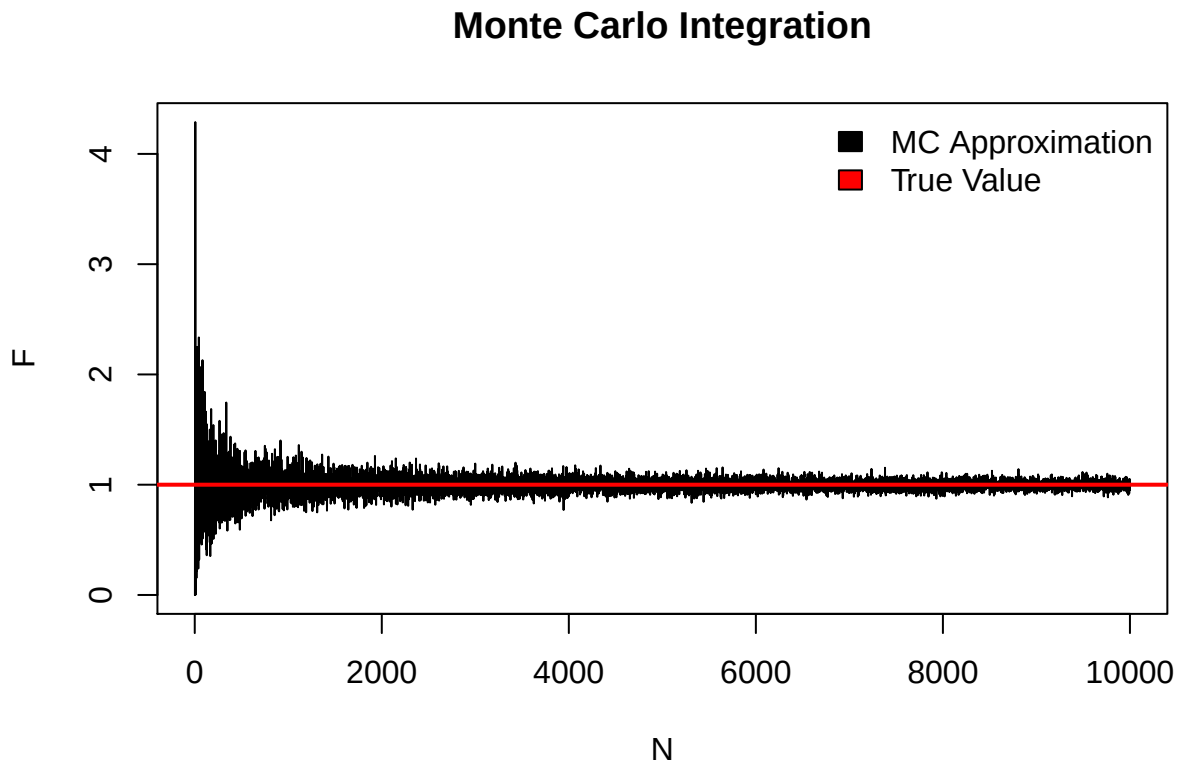
```
## [1] 0.9999999
```

```
## calculate the value using Monte Carlo algorithm
# Since the integration interval is infinite,
```

```r
# we can let t=1/(x+1) and then transfer the interval to [0,1].
MC <- function(n){
  t <- runif(n)
  f <- 1/t^2*(25*cos(1/t-1)*exp(-25*(1/t-1)))
  MC <- mean(f)
}

F <- rep(0,10^4)
for (i in 1:10^4){
  F[i] <- MC(i)
}

# Plot
plot(1:10^4, F, type='l', xlab="N", ylab="F", main="Monte Carlo Integration")
abline(h=mu$value, col="red", lwd=2)
legend('topright', c("MC Approximation","True Value"), fill=c("black", "red"), cex=1, bty='n')
```



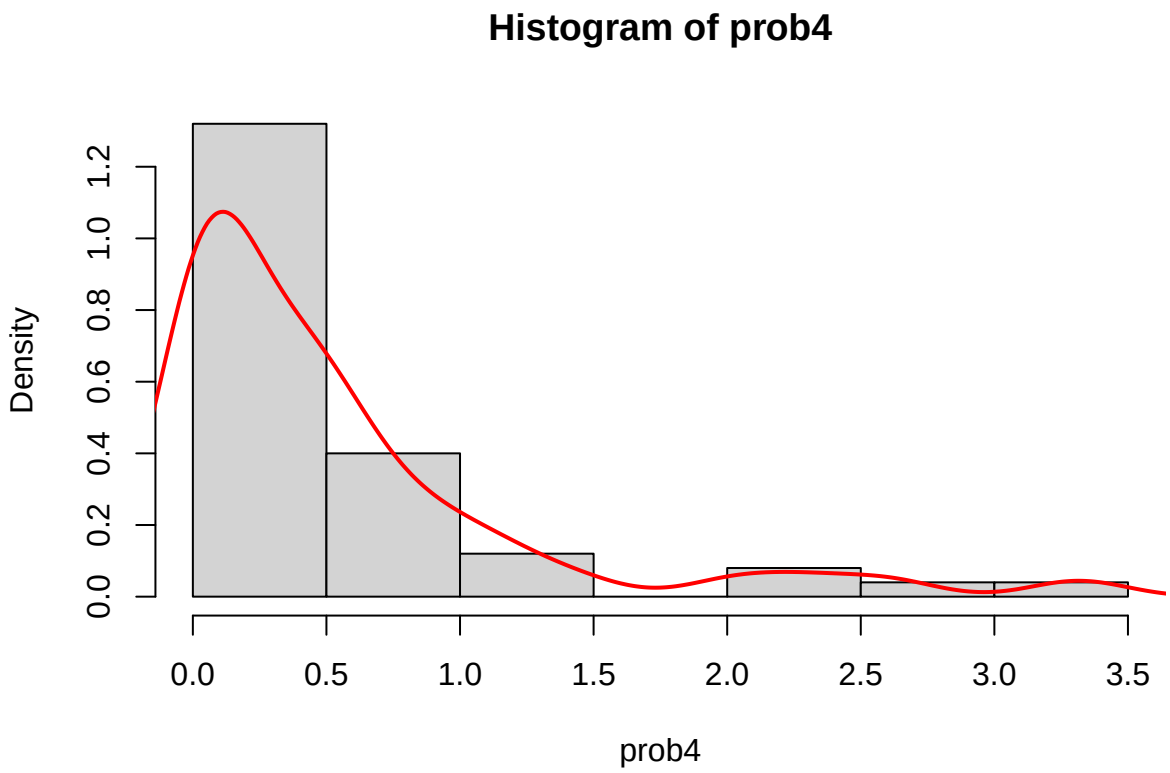

```r
# Access the error
error <- abs(mu$value - mean(F))
cat("The error is", error)
```

```
## The error is 0.001625055
```

Exercise 4

(a)

```
hist(prob4, probability = T)
lines(density(prob4), lwd=2, col='red')
```



(b)

According to the method of moment and the properties of gamma distribution, we can estimate the parameters by solving the following equations.

$$\hat{\mu} = E(X) = \frac{\alpha}{\beta}$$

$$\sigma^2 = E(x-\mu)^2 = \frac{\alpha}{\beta^2}$$

```
mean <- mean(prob4)
var <- var(prob4)
b <- mean/var
a <- mean^2/var
```

$\alpha = 0.568931$ and $\beta = 1.0602121$

(c)

```
# generate 1000 new samples
set.seed(2)
boot.sample <- list()
for (i in 1:1000){
  boot.sample[[i]] <- sample(prob4,size = 50, replace = TRUE)
}

# calculate parameters in each sample
parameters <- matrix(NA, 1000, 2, dimnames=list(NULL, c("alpha", "beta")))
for (i in 1:1000){
  parameters[i,2] <- mean(boot.sample[[i]]) / var(boot.sample[[i]]) #beta
  parameters[i,1] <- mean(boot.sample[[i]]) ^ 2 / var(boot.sample[[i]]) #alpha
}
boot.mean <- apply(parameters, 2, mean)
cat("The Bootstrap Mean:\n")
print(boot.mean)
boot.sd <- apply(parameters, 2, sd)
cat("\nThe Bootstrap Standard Deviation:\n")
print(boot.sd)
CI95 <- apply(parameters, 2, quantile, p=c(0.025, 0.975))
cat("\nThe Bootstrap 95% confidence interval:\n")
print(CI95)
```

```
## The Bootstrap Mean:

##     alpha      beta
## 0.6298554 1.2122591

##
## The Bootstrap Standard Deviation:

##     alpha      beta
## 0.1522462 0.4134740

##
## The Bootstrap 95% confidence interval:

##           alpha      beta
## 2.5%  0.3979654 0.7569899
## 97.5% 0.9933485 2.3269289
```

The results of parameters are almost the same with results in problem 4b..

Exercise 5.4.12

(a)

```
sample <- sample(1:10000, 500) #SRS without replacement
```

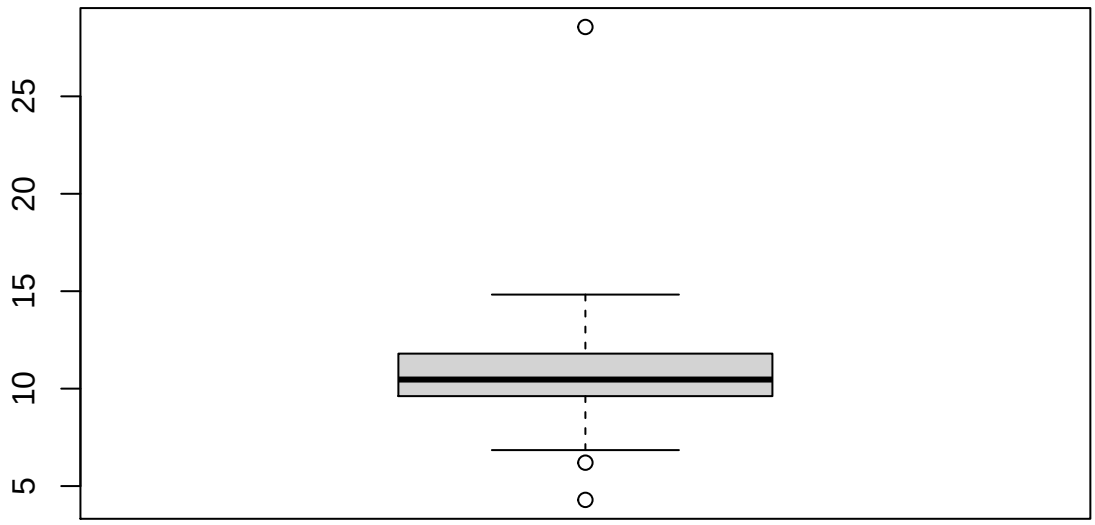
(b)

```
sample <- sample(1:10000, 500, replace=TRUE) #SRS with replacement
```

Exercise 5.5.18

(a)

```r
# Generate samples0
sample <- c(rnorm(30, 10, 2),rnorm(1, 30, 2))
#plot
boxplot(sample)
```



(b)

There are few outliers in the plot.

(c)

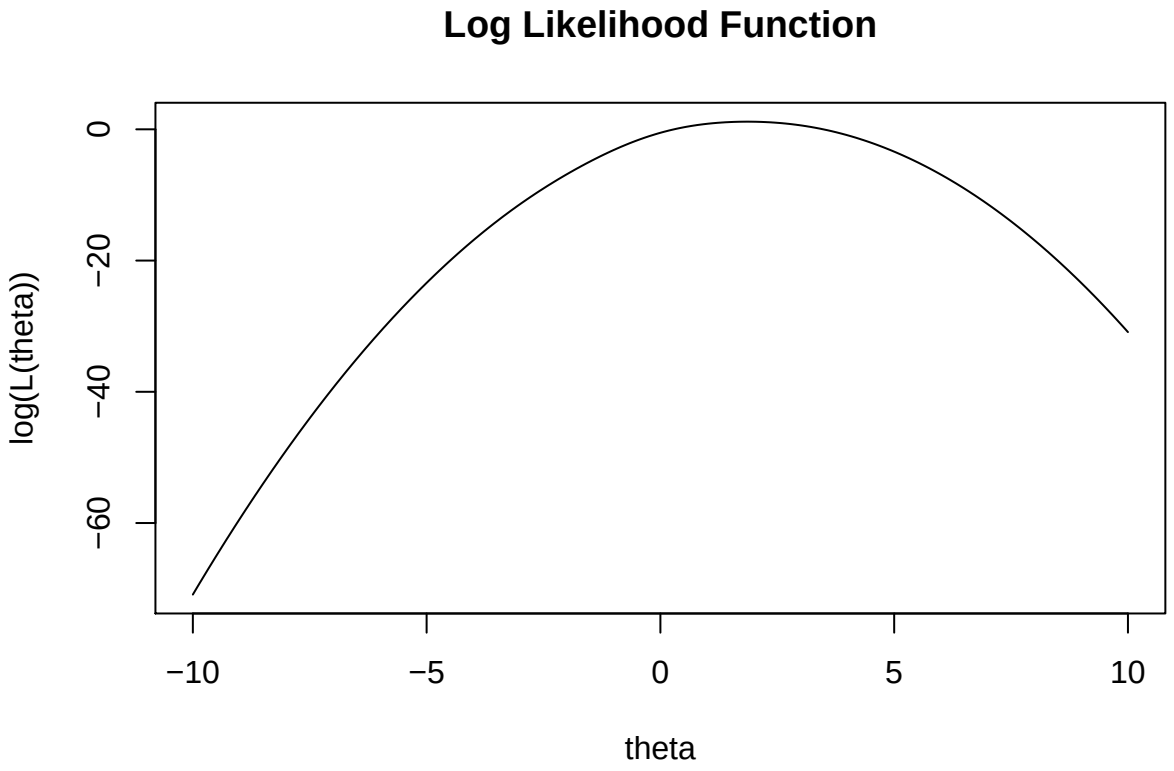
For data with extreme values, the median provides a better estimate of location than does the mean and IQR is more appropriate than standard deviation to measure the spread. It is because that extreme outliers would distort the mean and std.

Exercise 6.2.17

```
L<-function(theta){
    return((exp(-(theta - 1) ^ 2) / 2) + 3 * exp(-((theta - 2) ^ 2) / 2))
}
# MLE
theta <- seq(-10, 10, length=1000)
theta[which(L(theta) == max(L(theta)))]
```

```
## [1] 1.871872
```

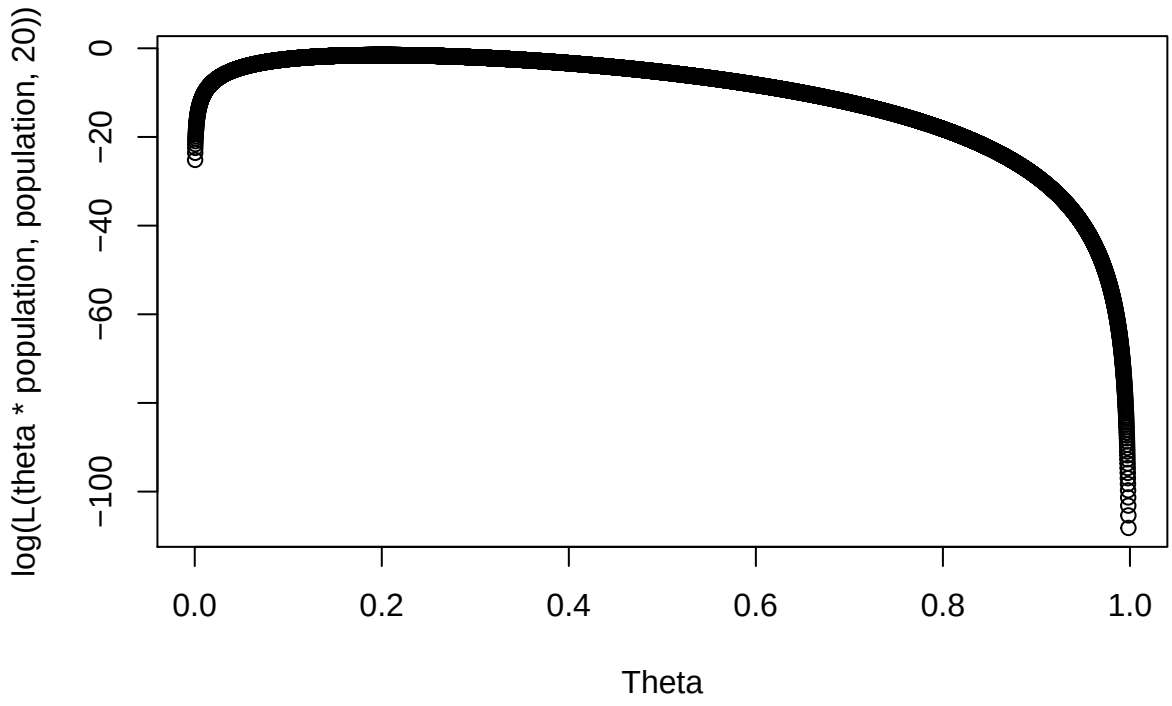
```
plot(theta, log(L(theta)), type='l', main="Log Likelihood Function")
```



Exercise 6.2.25

(a)

```
L <- function(left, population, sample){
    right <- population - left
    return(choose(left, 4) * choose(right, sample - 4) / choose(population, sample))
}
#MLE
population <- 10^4
theta <- (1:population) / population
plot(theta, log(L(theta * population, population,20)), xlab='Theta')
```

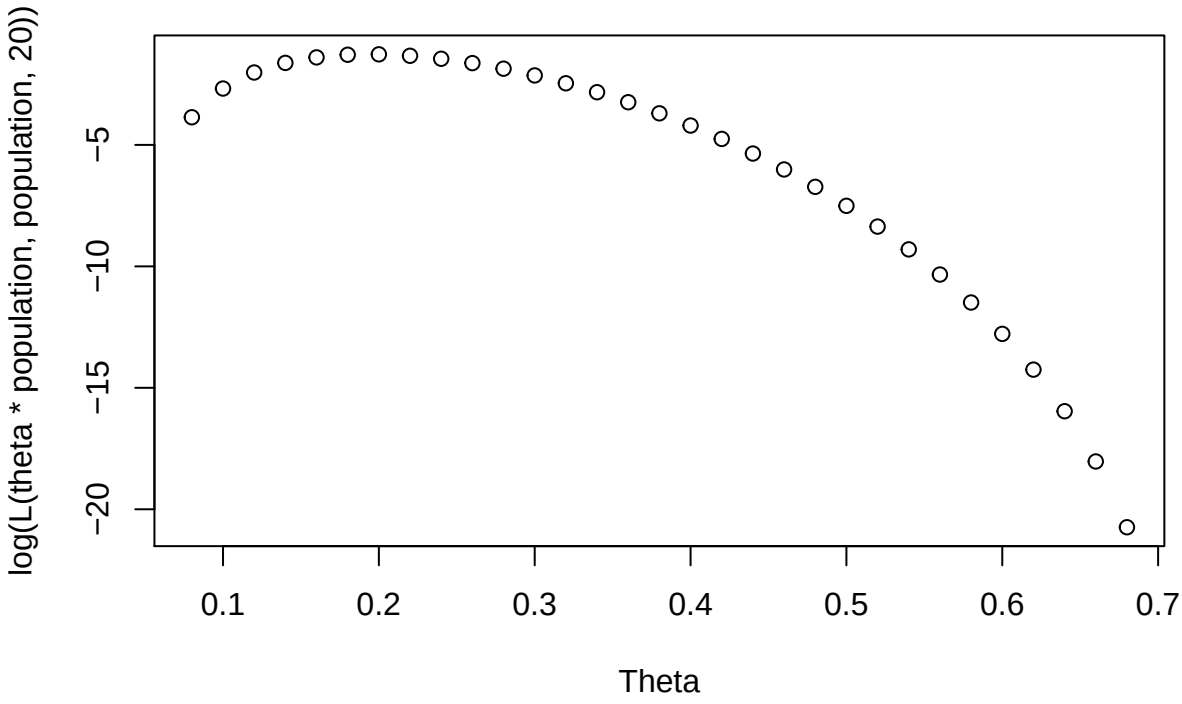



```
theta[which(log(L(theta * population, population,20)) == max(log(L(theta*population, population,20))))]
```

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

(b)

```
population <- 50
theta <- (4:34) / population
plot(theta, log(L(theta*population, population,20)), xlab='Theta')
```




```
theta[which(log(L(theta*population, population,20)) == max(log(L(theta*population, population,20))))]
```

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

Exercise 6.3.22

```
set.seed(1)
interval <- function(n){
    sample <- rnorm(n)
    if(mean(sample) - (sd(sample) / sqrt(n)) <= 0 && mean(sample) + (sd(sample) / sqrt(n)) >= 0){
        return(1)
    }else{
        return(0)
    }
}

#n=5
samples1 <- replicate(10^4, interval(5))
p1 <- sum(samples1) / 10^4

#n=10
samples2 <- replicate(10^4, interval(10))
p2 <- sum(samples2) / 10^4

#n=100
```

```r
samples3 <- replicate(10^4, interval(100))
p3 <- sum(samples3) / 10^4

print(c(p1,p2,p3))
```

```
## [1] 0.6329 0.6615 0.6833
```

The increase of sample size n is associated with the increase of proportion of times this interval contains μ. It is intuitive that the confidential interval is more efficient when it has a larger sample.

Exercise 6.4.17

```r
# data of 6.4.1
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, 0.15, -3.64, 4.88, 4.55)

# the parameters makes dataset X most likely is the mean and variance of X
# plug-in estimator:
estimator <- pnorm((3 - mean(X)) / sd(X))
cat("Using plug-in MLE estimator, we can get F(3)=", estimator)
```

```
## Using plug-in MLE estimator, we can get F(3)= 0.5133075
```

```r
#  bias
sample_proportion <- length(which(X < 3)) / length(X)
bias <- estimator - sample_proportion
mse <- bias^2 + var(X)
cat("The bias is", bias)
```

```
## The bias is 0.1133075
```

```r
# Bootstrap:
set.seed(1)
estimator_f <- function(X, value=3) {
  return(pnorm((value - mean(X)) / sd(X)))
}

proportion_f <- function(X, value=3) {
  return(length(which(X < value)) / length(X))
}

bootstrap <- function(n) {
  samples_matrix <- matrix(sample(X, length(X) * n, replace = T), ncol = length(X))
  estimator <- apply(samples_matrix, 1, estimator_f)
  proportion <- apply(samples_matrix, 1, proportion_f)
  bias <- mean(estimator) - mean(proportion)
  MSE <- bias^2 + var(estimator)
  return(MSE)
}

MSE_10e3 <- bootstrap(10^3) # 10^3
MSE_10e4 <- bootstrap(10^4) # 10^4
print(c(MSE_10e3, MSE_10e4))
```

```
## [1] 0.02035768 0.02028708
```