# CS 6313.001

Statistical Methods for Data Science

By

Prof. Min Chen

# MINI PROJECT-3

# DUO GROUP-41


1.Amit Kumar AXK210047

2.Vedant Paresh Shah VXS200021

**Contribution of each group member:**

Both worked together and finished the questions as instructed. First went through all the details required, followed the class Note and textbook, learned R then wrote down the scripts. Both of us worked efficiently to complete the required project and finished it on time.

**1. (8 points) Suppose we would like to estimate the parameter θ (> 0) of a Uniform (0, θ) population based on a random sample X1, . . . , Xn from the population. In the class, we have discussed two estimators for θ — the maximum likelihood estimator, ˆθ1 = X(n) , where X(n) is the maximum of the sample, and the method of moments estimator, ˆθ2 = 2X, where X is the sample mean. The goal of this exercise is to compare the mean squared errors of the two estimators to determine which estimator is better. Recall that the mean squared error of an estimator ˆθ of a parameter θ is defined as E{( ˆθ − θ) 2}. For the comparison, we will focus on n = 1, 2, 3, 5, 10, 30 and θ = 1, 5, 50, 100.**

**(a) Explain how you will compute the mean squared error of an estimator using Monte Carlo simulation.**

**Answer:**

We will generate a sample of size n from the uniform distribution in the interval (0, θ). Then we will replicate this step x times and store the values of the x samples. Now using these values of x samples, we will calculate the estimator value. The mean squared error is nothing, but the estimated value difference is squared between the estimator and parameter.

**(b) For a given combination of (n, θ), compute the mean squared errors of both ˆθ1 and ˆθ2 using Monte Carlo simulation with N = 1000 replications. Be sure to compute both estimates from the same data.**

**Answer:**

Here we have created a function named estimate(n, theta, x) where the value of n and theta is given to and x is the number of times the simulation needs to be performed. The function will return value in the array with 2 values first of MLE MSE and second of MOM MSE. As we have to run simulation 1000 times we will put the value of x=1000. And then we will put value of n and theta as given in the above question. The estimate function will calculate the mean squared error using the formula E{($\theta$ - $\theta$)^2} and it will return the mean squared errors for both estimators in an array format.

Let us take the first combination of n and $\theta$ as (1, 1) so we get the mean squared errors as:

$MSE(\theta 1)$ = 0.3424121 $MSE(\theta 2)$ = 0.3314814

```
> estimate(1,1,1000)
[1] 0.3424121 0.3314814
```

**Code Snippet:**

```r
estimate <- function(n,theta,x){

    data <- replicate(x,runif(n,0,theta))
    #MLE
    if(NCOL(data)!=1)
    {
      MLE <- apply(data,2,max)
    }
    else
    {
      MLE<-data
    }
    MLEresult <- sum((MLE-theta)^2)/x
    #MOM
    if(NCOL(data)!=1)
    {
      MOM <- apply(data,2,mean)
    }
    else
    {
      MOM<-2*data
    }
    MOMresult <- sum((MOM-theta)^2)/x
  return(c(MLEresult,MOMresult))
}
```

**(c) Repeat (b) for the remaining combinations of (n, θ). Summarize your results graphically.**

**Answer:**

Here we will repeat above code of estimate and we will try all the values of n for all values of theta.

```
> #n=1
> estimate(1,1,1000)
[1] 0.3307547 0.3336730
> estimate(1,10,1000)
[1] 33.87812 33.19959
> estimate(1,50,1000)
[1] 823.6242 842.2511
> estimate(1,100,1000)
[1] 3396.674 3286.676
```

```
> #n=2
> estimate(2,1,1000)
[1] 0.1584307 0.2823891
> estimate(2,10,1000)
[1] 16.71698 29.21377
> estimate(2,50,1000)
[1] 412.6531 726.0084
> estimate(2,100,1000)
[1] 1670.994 2871.608
```

```
> #n=3
> estimate(3,1,1000)
[1] 0.1015574 0.2735926
> estimate(3,10,1000)
[1] 10.58683 28.04506
> estimate(3,50,1000)
[1] 229.9007 669.7272
> estimate(3,100,1000)
[1]  979.3194 2759.6709
```

```
> #n=5
> estimate(5,1,1000)
[1] 0.05174727 0.27228723
> estimate(5,10,1000)
[1]  4.508769 26.527940
> estimate(5,50,1000)
[1] 113.8885 656.4751
> estimate(5,100,1000)
[1]  505.594 2712.582
```

```
> #n=10
> estimate(10,1,1000)
[1] 0.01366183 0.25625570
> estimate(10,10,1000)
[1]  1.524328 26.048655
> estimate(10,50,1000)
[1]  38.67074 644.96876
> estimate(10,100,1000)
[1]  141.4857 2585.2164
```

```
> #n=30
> estimate(30,1,1000)
[1] 0.001853629 0.251036930
> estimate(30,10,1000)
[1]  0.2033139 25.1648937
> estimate(30,50,1000)
[1]   4.42575 629.89270
> estimate(30,100,1000)
[1]   21.78003 2526.46104
```

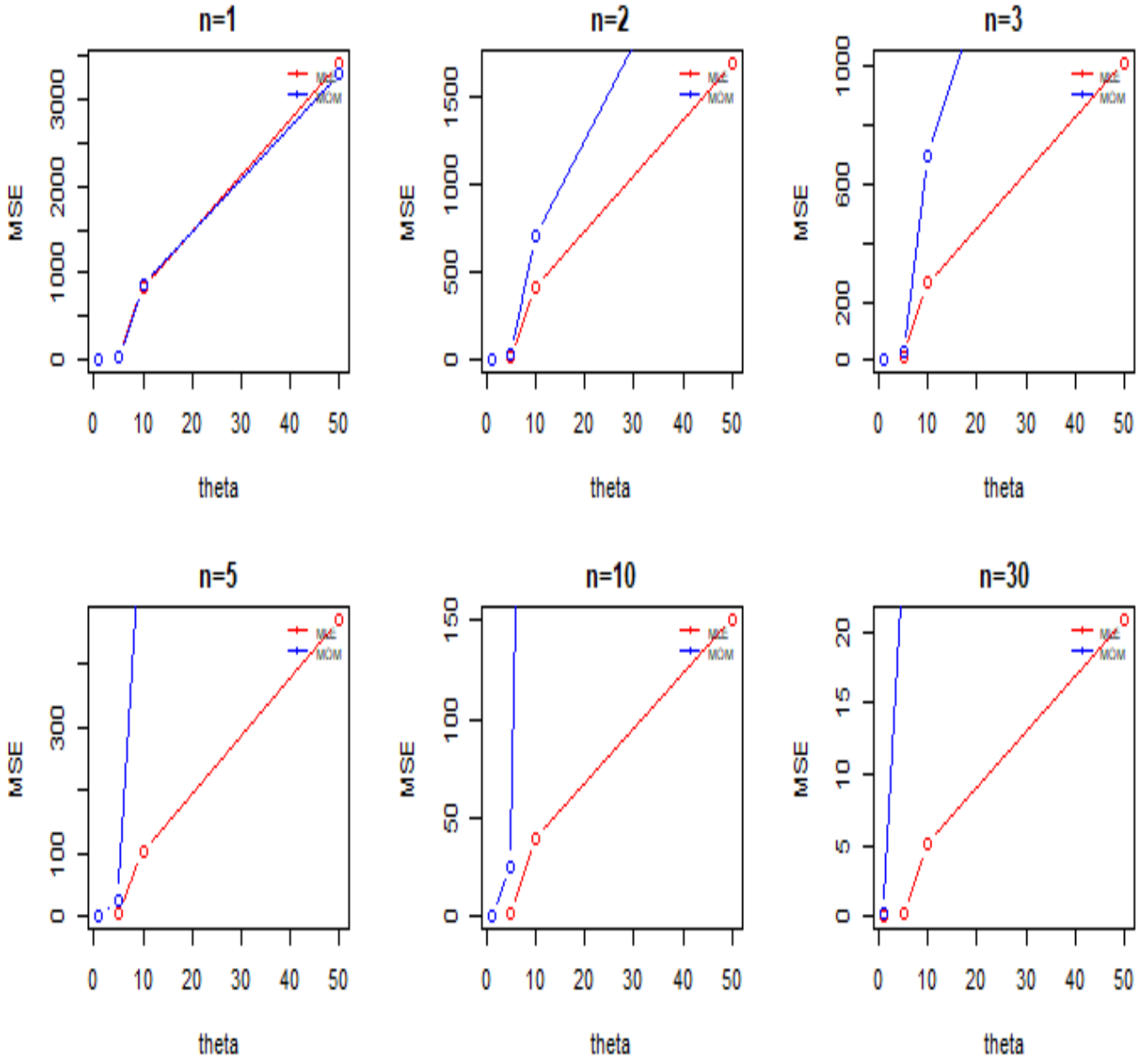
We will now plot these values in graphs and we will use par() function that will accommodate graph in matrix form.

## Code-Snipet:

```
# Plotting graphs where value of n is fixed and value of theta is varying.

par(mfrow=c(2,3))

theta<-c(1,5,10,50)

n1_1=c(estimate(1,1,1000)[1],estimate(1,10,1000)[1],estimate(1,50,1000)[1],estimate(1,100,1000)[1])
n1_2=c(estimate(1,1,1000)[2],estimate(1,10,1000)[2],estimate(1,50,1000)[2],estimate(1,100,1000)[2])

n2_1=c(estimate(2,1,1000)[1],estimate(2,10,1000)[1],estimate(2,50,1000)[1],estimate(2,100,1000)[1])
n2_2=c(estimate(2,1,1000)[2],estimate(2,10,1000)[2],estimate(2,50,1000)[2],estimate(2,100,1000)[2])

n3_1=c(estimate(3,1,1000)[1],estimate(3,10,1000)[1],estimate(3,50,1000)[1],estimate(3,100,1000)[1])
n3_2=c(estimate(3,1,1000)[2],estimate(3,10,1000)[2],estimate(3,50,1000)[2],estimate(3,100,1000)[2])

n5_1=c(estimate(5,1,1000)[1],estimate(5,10,1000)[1],estimate(5,50,1000)[1],estimate(5,100,1000)[1])
n5_2=c(estimate(5,1,1000)[2],estimate(5,10,1000)[2],estimate(5,50,1000)[2],estimate(5,100,1000)[2])

n10_1=c(estimate(10,1,1000)[1],estimate(10,10,1000)[1],estimate(10,50,1000)[1],estimate(10,100,1000)[1])
n10_2=c(estimate(10,1,1000)[2],estimate(10,10,1000)[2],estimate(10,50,1000)[2],estimate(10,100,1000)[2])

n30_1=c(estimate(30,1,1000)[1],estimate(30,10,1000)[1],estimate(30,50,1000)[1],estimate(30,100,1000)[1])
n30_2=c(estimate(30,1,1000)[2],estimate(30,10,1000)[2],estimate(30,50,1000)[2],estimate(30,100,1000)[2])
```

```
plot(theta,n1_1,ylab="MSE",type="b",xlab="theta",main="n=1",col="red")
lines(theta,n1_2,col="blue",type="b")
legend("topright",legend=c("MLE", "MOM"),col=c('red','blue'),text.col=c('black','black'),
       lty=1,pch=1,inset=0.01,ncol=1,cex=0.6,bty = 'n')

plot(theta,n2_1,ylab="MSE",type="b",xlab="theta",main="n=2",col="red")
lines(theta,n2_2,col="blue",type="b")
legend("topright",legend=c("MLE", "MOM"),col=c('red','blue'),text.col=c('black','black'),
       lty=1,pch=1,inset=0.01,ncol=1,cex=0.6,bty = 'n')

plot(theta,n3_1,ylab="MSE",type="b",xlab="theta",main="n=3",col="red")
lines(theta,n3_2,col="blue",type="b")
legend("topright",legend=c("MLE", "MOM"),col=c('red','blue'),text.col=c('black','black'),
       lty=1,pch=1,inset=0.01,ncol=1,cex=0.6,bty = 'n')

plot(theta,n5_1,ylab="MSE",type="b",xlab="theta",main="n=5",col="red")
lines(theta,n5_2,col="blue",type="b")
legend("topright",legend=c("MLE", "MOM"),col=c('red','blue'),text.col=c('black','black'),
       lty=1,pch=1,inset=0.01,ncol=1,cex=0.6,bty = 'n')

plot(theta,n10_1,ylab="MSE",type="b",xlab="theta",main="n=10",col="red")
lines(theta,n10_2,col="blue",type="b")
legend("topright",legend=c("MLE", "MOM"),col=c('red','blue'),text.col=c('black','black'),
       lty=1,pch=1,inset=0.01,ncol=1,cex=0.6,bty = 'n')

plot(theta,n30_1,ylab="MSE",type="b",xlab="theta",main="n=30",col="red")
lines(theta,n30_2,col="blue",type="b")
legend("topright",legend=c("MLE", "MOM"),col=c('red','blue'),text.col=c('black','black'),
       lty=1,pch=1,inset=0.01,ncol=1,cex=0.6,bty = 'n')
```

**Output:**



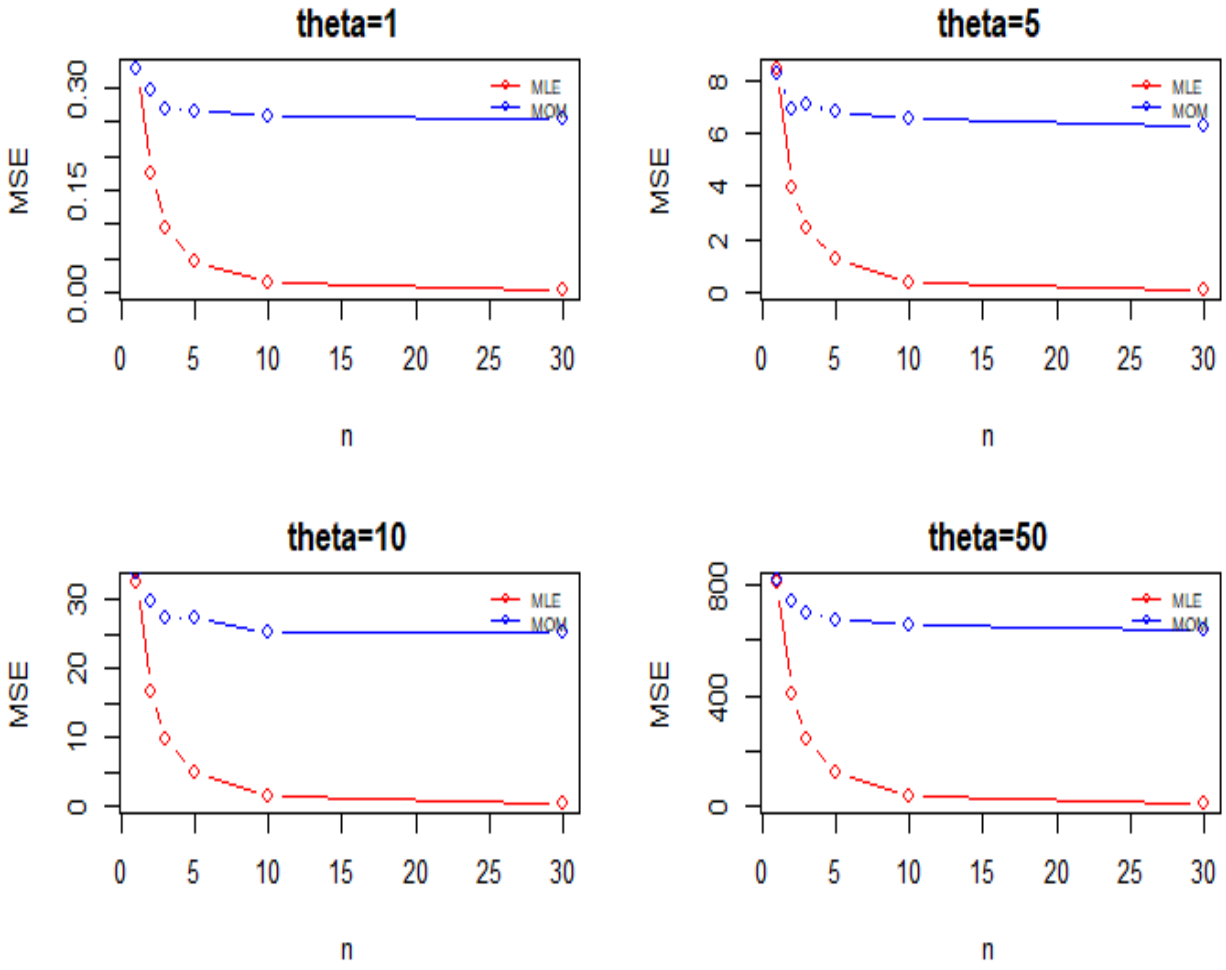

**Graph 1: Mean squared errors of MLE and MOM, $\theta$ with fixed n**

## Code-Snippet:

```
# Plotting graphs where value of theta is fixed and value of n is varying.

par(mfrow=c(2,2))

n<-c(1,2,3,5,10,30)

theta1_1=c(estimate(1,1,1000)[1],estimate(2,1,1000)[1],estimate(3,1,1000)[1],
         estimate(5,1,1000)[1],estimate(10,1,1000)[1],estimate(30,1,1000)[1])
theta1_2=c(estimate(1,1,1000)[2],estimate(2,1,1000)[2],estimate(3,1,1000)[2],
         estimate(5,1,1000)[2],estimate(10,1,1000)[2],estimate(30,1,1000)[2])

theta5_1=c(estimate(1,5,1000)[1],estimate(2,5,1000)[1],estimate(3,5,1000)[1],
         estimate(5,5,1000)[1],estimate(10,5,1000)[1],estimate(30,5,1000)[1])
theta5_2=c(estimate(1,5,1000)[2],estimate(2,5,1000)[2],estimate(3,5,1000)[2],
         estimate(5,5,1000)[2],estimate(10,5,1000)[2],estimate(30,5,1000)[2])

theta10_1=c(estimate(1,10,1000)[1],estimate(2,10,1000)[1],estimate(3,10,1000)[1],
          estimate(5,10,1000)[1],estimate(10,10,1000)[1],estimate(30,10,1000)[1])
theta10_2=c(estimate(1,10,1000)[2],estimate(2,10,1000)[2],estimate(3,10,1000)[2],
          estimate(5,10,1000)[2],estimate(10,10,1000)[2],estimate(30,10,1000)[2])

theta50_1=c(estimate(1,50,1000)[1],estimate(2,50,1000)[1],estimate(3,50,1000)[1],
          estimate(5,50,1000)[1],estimate(10,50,1000)[1],estimate(30,50,1000)[1])
theta50_2=c(estimate(1,50,1000)[2],estimate(2,50,1000)[2],estimate(3,50,1000)[2],
          estimate(5,50,1000)[2],estimate(10,50,1000)[2],estimate(30,50,1000)[2])
```

```
plot(n,theta1_1,ylab="MSE",type="b",xlab="n",main="theta=1",col="red")
lines(n,theta1_2,col="blue",type="b")
legend("topright",legend=c("MLE", "MOM"),col=c('red','blue'),text.col=c('black','black'),
       lty=1,pch=1,inset=0.01,ncol=1,cex=0.6,bty = 'n')

plot(n,theta5_1,ylab="MSE",type="b",xlab="n",main="theta=5",col="red")
lines(n,theta5_2,col="blue",type="b")
legend("topright",legend=c("MLE", "MOM"),col=c('red','blue'),text.col=c('black','black'),
       lty=1,pch=1,inset=0.01,ncol=1,cex=0.6,bty = 'n')

plot(n,theta10_1,ylab="MSE",type="b",xlab="n",main="theta=10",col="red")
lines(n,theta10_2,col="blue",type="b")
legend("topright",legend=c("MLE", "MOM"),col=c('red','blue'),text.col=c('black','black'),
       lty=1,pch=1,inset=0.01,ncol=1,cex=0.6,bty = 'n')

plot(n,theta50_1,ylab="MSE",type="b",xlab="n",main="theta=50",col="red")
lines(n,theta50_2,col="blue",type="b")
legend("topright",legend=c("MLE", "MOM"),col=c('red','blue'),text.col=c('black','black'),
       lty=1,pch=1,inset=0.01,ncol=1,cex=0.6,bty = 'n')
```

**Output:**




**Graph 2: Mean squared errors of MLE and MOM, n with fixed $\theta$**

**(d) Based on (c), which estimator is better? Does the answer depend on n or θ? Explain. Provide justification for all your conclusions**.

**Answer:**

From above question (c) graphs we can observe in Graph 2 that no matter what the value of theta is we get almost same graph for all values of n. So, we can determine that value of theta does not effect the value of estimator. So, value of n does have affect on the answer.

In Graph 1 we can the Mean squared error values plotted where value of theta is varying and value n fixed.It is very evident that we can use Method of Moments estimator (MOM) for small values of n = 1, 2 . But as the n = 3, 5, 10, 30 value increases, the Maximum Likelihood Estimator (MLE) is better. For larger values of n, MLE is better and as n increases MLE would be the preferred choice. MLE is preferred because the mean squared error is less for the same value of n in comparison to MOM.

Thus, The value of answer depend on n and MLE(Maximum Likelihood Estimator Method) is better estimator method.

**2. (12 points) Suppose the lifetime, in years, of an electronic component can be modeled by a continuous random variable with probability density function**

$$f(x) = \begin{cases} \frac{\theta}{x^{\theta+1}} & x \geq 1, \\ 0, & x < 1, \end{cases}$$

**where θ > 0 is an unknown parameter. Let X1, . . . , Xn be a random sample of size n from this population.**

**(a) Derive an expression for maximum likelihood estimator of θ.**

**Answer:** As given in the question, probability density function is:

$$f(x) = \begin{cases} \frac{Q}{x^{Q+1}} ; & x \geq 1, \\ 0 & ; x < 1, \end{cases}$$

where θ > 0 is an unknown parameter. Let X1,……, Xn be a random sample of size n from this population.

Likelihood function can be written as: $L(\theta) = \prod_{i=1}^{n} (\theta / x_i^{\theta+1})$ ----------(1)

Now take the log on both side of the equation (1),

$log(L(\theta)) = log(\prod_{i=1}^{n} (\theta / x_i^{\theta+1})$

$= log\ (\theta^n * \prod_{i=1}^{n} (1/x_i^{\theta+1})$

$= n\ log\theta + \sum_{i=1}^{n} log\ (x_i^{-\theta-1})$

$= n\ log\theta - (\theta + 1)\ \sum_{i=1}^{n} \log\ (x_i)$

$= n\ log\theta - \theta\ \sum_{i=1}^{n} \log\ (x_i) - \sum_{i=1}^{n} \log\ (x_i)$ ----------(2)

Now partially differentiate the equation (2) and quate it to 0

$$\frac{\partial(log(L(\theta)))}{\partial\theta} = 0$$

$$=> \frac{n}{\theta} - \sum_{i=1}^{n} \log(x_i) = 0$$

$$=> \frac{n}{\theta} = \sum_{i=1}^{n} \log(x_i)$$

$$\hat{\theta}mle = \frac{n}{\sum_{i=1}^{n} \log(x_i)}$$

**(b) Suppose n = 5 and the sample values are x1 = 21.72, x2 = 14.65, x3 = 50.42, x4 = 28.78, x5 = 11.23. Use the expression in (a) to provide the maximum likelihood estimate for θ based on these data.**

**Answer:** To find the maximum likelihood estimate for Q based on the data given, we will have to put the values into the equation derived in 1(a) i.e.

$$\hat{\theta}mle = \frac{n}{\sum_{i=1}^{n} \log(x_i)}$$

$$\hat{\theta}mle = \frac{5}{\log(21.72)+\log(14.65)+\log(50.42)+\log(28.78)+\log(11.23)}$$

$$\hat{\theta}mle = \frac{5}{\log(21.72*14.65*50.42*28.78*11.23)}$$

$$\hat{\theta}mle = \frac{5}{\log(5185263.52319)}$$

$$\hat{\theta}mle = \frac{5}{\log(5185263.52319)}$$

$$\hat{\theta}mle = \frac{5}{15.46}$$

$$\boxed{\hat{\theta}mle = 0.3234}$$

**(c) Even though we know the maximum likelihood estimate from (b), use the data in (b) to obtain the estimate by numerically maximizing the log-likelihood function using optim function in R. Do your answers match?**

**Answer:**

By default, optim function searches for parameters, which minimise the function fn. In order to find a maximum, we will have to change the sign of the the returned value.

**R-Code to maximizing the log-likelihood function**

```
dataSet <- c(21.72, 14.65, 50.42, 28.78, 11.23)

ngtv.lglklihood.fn <- function(par, dat) {
        result = length(dat) * log(par) - (par + 1) * sum(log(dat))
        result
        return (-result)
}

# Here par is the Initial values for the parameters to be optimized over, dat is the dataset provided in 2(b)

# (-) sign is added while returning the result to find the maximum

mleValue <- optim(par=0.926, fn=ngtv.lglklihood.fn, method="L-BFGS-B", hessian=TRUE, lower=0.01, dat=dataSet)

# Here fn is the function being minimized

mleValue$par

[1] 0.3233885
```

**R-Code Snippet:**

```
> dataSet <- c(21.72, 14.65, 50.42, 28.78, 11.23)
> ngtv.lglklihood.fn <- function(par, dat) {
+   result = length(dat) * log(par) - (par + 1) * sum(log(dat))
+   result
+   return (-result)
+ }
> mleValue <- optim(par=0.926, fn=ngtv.lglklihood.fn, method="L-BF
S-B", hessian=TRUE,
+                   lower=0.01, dat=dataSet)
> mleValue$par
[1] 0.3233885
```

In 2(b) our value of $\theta mle$= 0.3234, the value obtained using R-Code is also 0.32348 ≈ 0.3234. Here the $\theta mle$ is same using both the method.

**(d) Use the output of numerical maximization in (c) to provide an approximate standard error of the maximum likelihood estimate and an approximate 95% confidence interval for θ. Are these approximations going to be good? Justify your answer.**

**Answer:**

**R-Code:**

```
# Finding the standard error
standardError <- (1/mleValue$hessian)^0.5
standardError
```

standardError value is: **0.1446223**

From R-code we get the Standard Error as: **0.1446223**

Also, it is given that, confidence interval for θ: 95%

$1 - \alpha = 0.95$

$=> \alpha = 1 - 0.95$

$=> \alpha = 0.05$

Hence, $\frac{\alpha}{2} = 0.05/2 = 0.025$

So, $1 - \frac{\alpha}{2} = 0.975$

```
#Confidence interval code
ci <- mleValue$par + c (-1, 1) * standardError * qnorm (0.975)
```

ci value is: **0.0399339 0.6068430**

Using R, we find out our confidence interval as: **(0.0399339, 0.6068430)**

**Code Snippet:**

```
> standardError <- (1/mleValue$hessian)^0.5
> standardError
           [,1]
[1,] 0.1446223
> #Confidence interval code
> ci <- mleValue$par + c(-1, 1) * standardError *qnorm(0.975)
```

From the confidence interval which we calculated , it can be inferred that the CI range

(0.0399, 0.6068) will capture the value of θ for 95% of the times. In the part b and c the calculated value of θ lies well within this range. Hence it proves that, these approximations will be good in determining the value of θ.