MINI PROJECT 3
GROUP NO. 28

Members
JEEVAN DSOUZA
SAMARTH SAIRAM

Contributions Rendered

Both the team members rendered contributions equally with respect to analysis and design of the solution for the problem statement and programming for the same.

1. a) #using Monte Carlo method

we first have to set the population parameter that is theta

Next we need to generate the samples from the population and need to calculate cap theta

Then we can get the MSE which is the squared difference of the population parameter theta and calculated cap theta.

1. b)

#Trying to calculate theta1 which is from MLE and Theta 2 that is method of moments estimator

#First we need to calculate both mle and moments from the same sample and then we need calculate MSE for both of them and we see in

```
MSE_moments_Esimate = function(sample_size,theta_value){

#we use Monte carlo simulation thus we use runif function

monte_carlo_value = runif(sample_size,min=0,max=theta_value)

#MLE calucation is obtained by taking the derivative of the log function and equating it to zero , but in here it is maximum of sample

Estimated_MLE = max(monte_carlo_value)

#Method of moments is 2 times the mean of generated value

Estimated_Moments = 2 * mean(monte_carlo_value)

#Next we return both the generated values

return (c(Estimated_MLE,Estimated_Moments))

}
```

```
#we need to calculate if for 1000 replications

MSE_moments_Calculte = function(sample_size,theta_value){

#We need to calculate MLE for both the estimators, so we call the function to estimate cap theta and and apply the formula to calculate both MLE and moments

Cap_Theta = replicate(1000,MSE_moments_Esimate(sample_size,theta_value))

#we use the definition of MSE to calculate the mean square difference

Mean_Sqaured_Error = (theta_value - Cap_Theta) ^ 2

#To get different values for both of the estimators

Moments_Calculated = Mean_Sqaured_Error[c(TRUE,FALSE)]

MLE_Calculated = Mean_Sqaured_Error[c(FALSE,TRUE)]

Moments_Calculated = mean(Moments_Calculated)

MLE_Calculated = mean(MLE_Calculated)

return(c(MLE_Calculated,Moments_Calculated))

}

#For sample n=1 and theta = 1

MSE_Calulate_1_1 = MSE_moments_Calculte(1,1)

MSE_Calulate_1_1
```

1.c)

```
#To calculate for all other combinations
```

#we will have 2 values one for MLE and the other for moments we need to plot them we will have 2 variations fixed value theta and

#constant n and vice versa

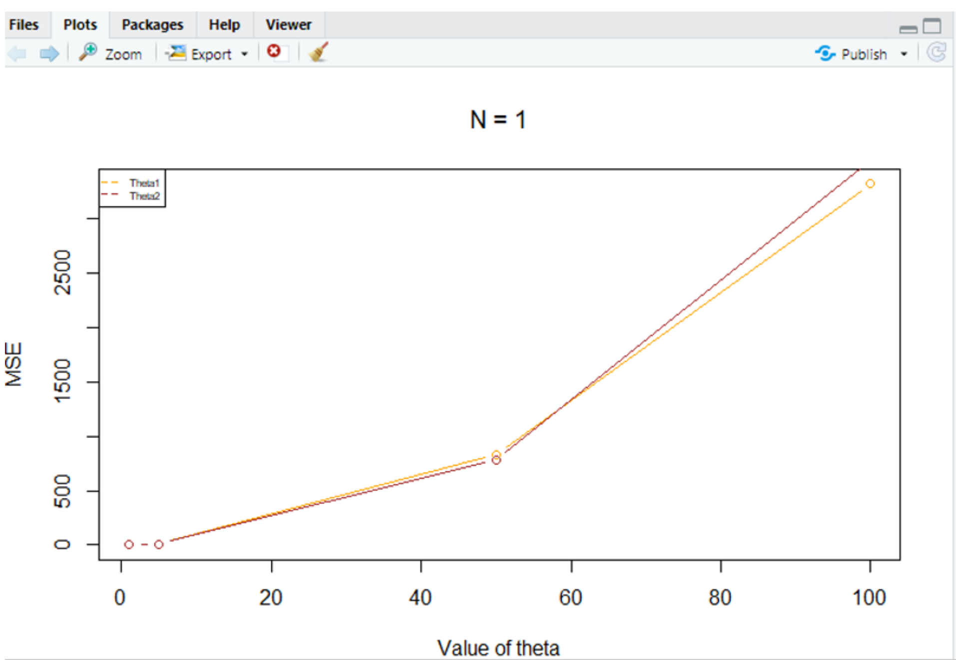
#First we add values of theta and n to a column then calculate the values by looping through 2 times and estimating those values

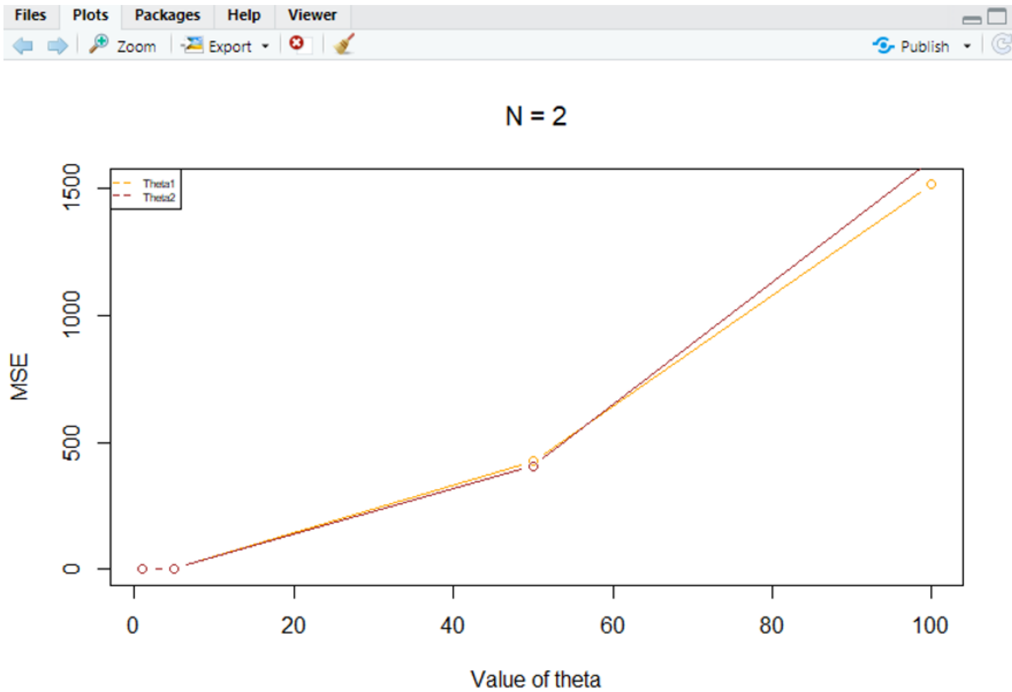
#This is for n

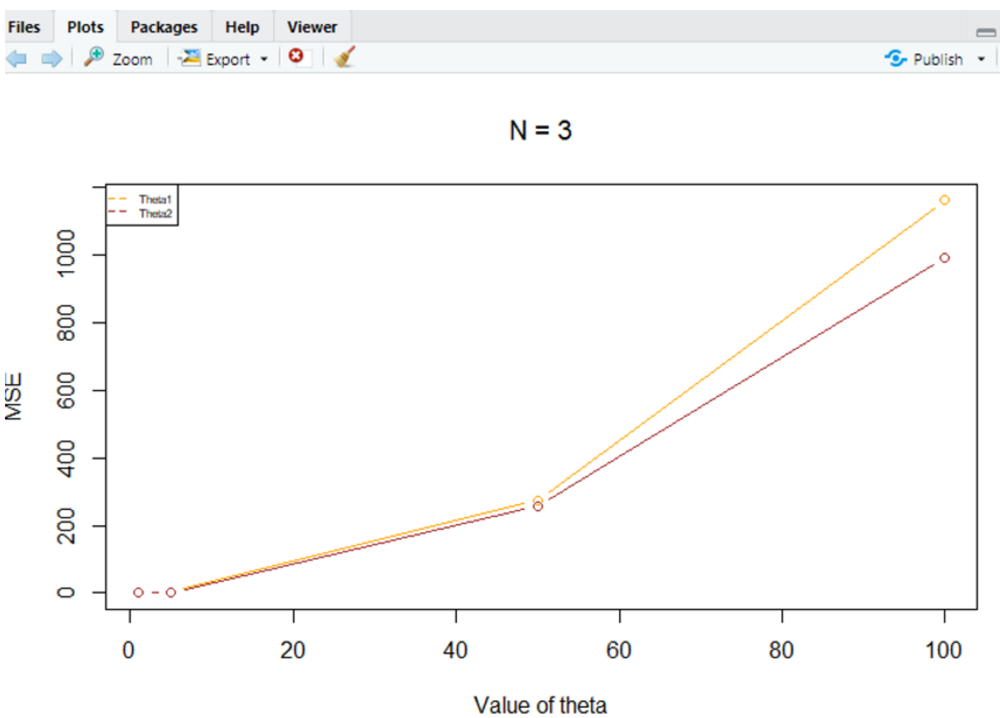
```
vals_of_n = c(1,2,3,5,10,30)

vals_of_theta = c(1,5,50,100)

counter = 0

#initializing both theta and

MSE_cal_theta1 = c(0,0,0,0)

MSE_Calc_theta2 = c(0,0,0,0)

for(itr1 in vals_of_n)

{

  counter=1

  for(itr2 in vals_of_theta)

  {

    calc = MSE_moments_Calculte(itr1,itr2)

    MSE_cal_theta1[counter] = calc[1]

    MSE_Calc_theta2[counter] = calc[2]

    counter = counter + 1

  }
```

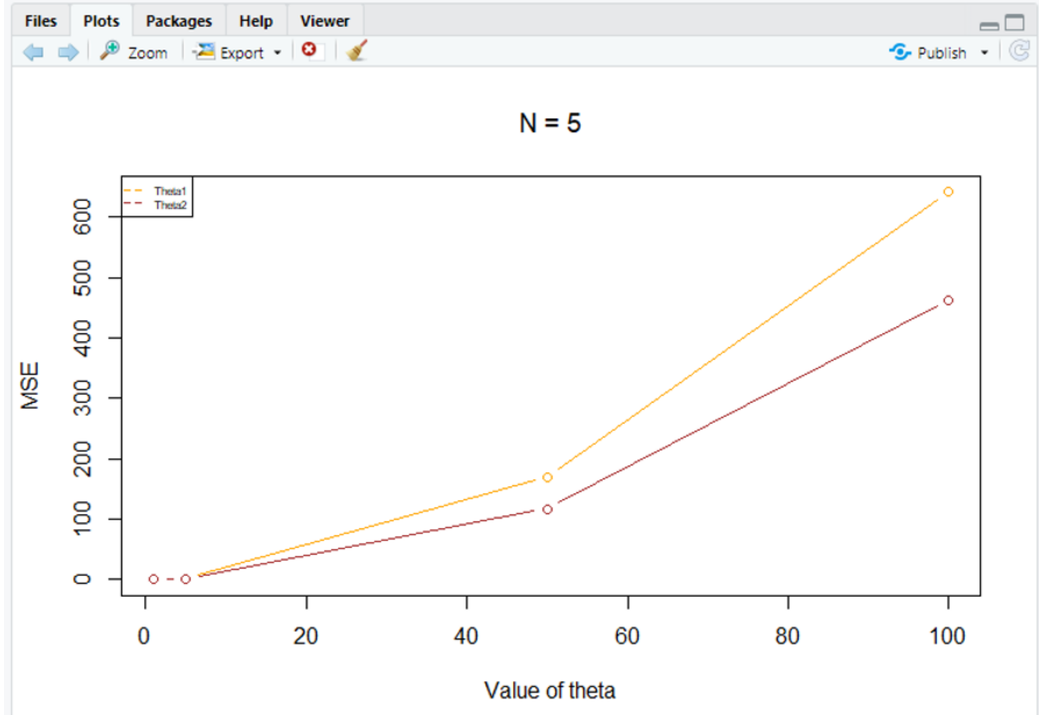
```
  plot(vals_of_theta,MSE_cal_theta1,ylab = 'MSE',main=bquote(paste("N = ", .
(itr1))), xlab = 'Value of theta',
       type = 'b',col='orange')
  lines(vals_of_theta,MSE_Calc_theta2,col='brown',type = 'b')
  legend("topleft",legend=c("Theta1","Theta2"),col=c('orange','brown'),cex =
0.5,lty=c(2,2),merge = TRUE)
}

#This is for theta values

vals_of_n = c(1,2,3,5,10,30)
vals_of_theta = c(1,5,50,100)
counter = 0
#initializing both theta and
MSE_cal_theta1 = c(0,0,0,0,0,0)
MSE_Calc_theta2 = c(0,0,0,0,0,0)

for(itr1 in vals_of_theta)
{
  counter=1
  for(itr2 in vals_of_n)
  {
    calc = MSE_moments_Calculte(itr2,itr1) #Interchanging the values in here
    MSE_cal_theta1[counter] = calc[1]
```

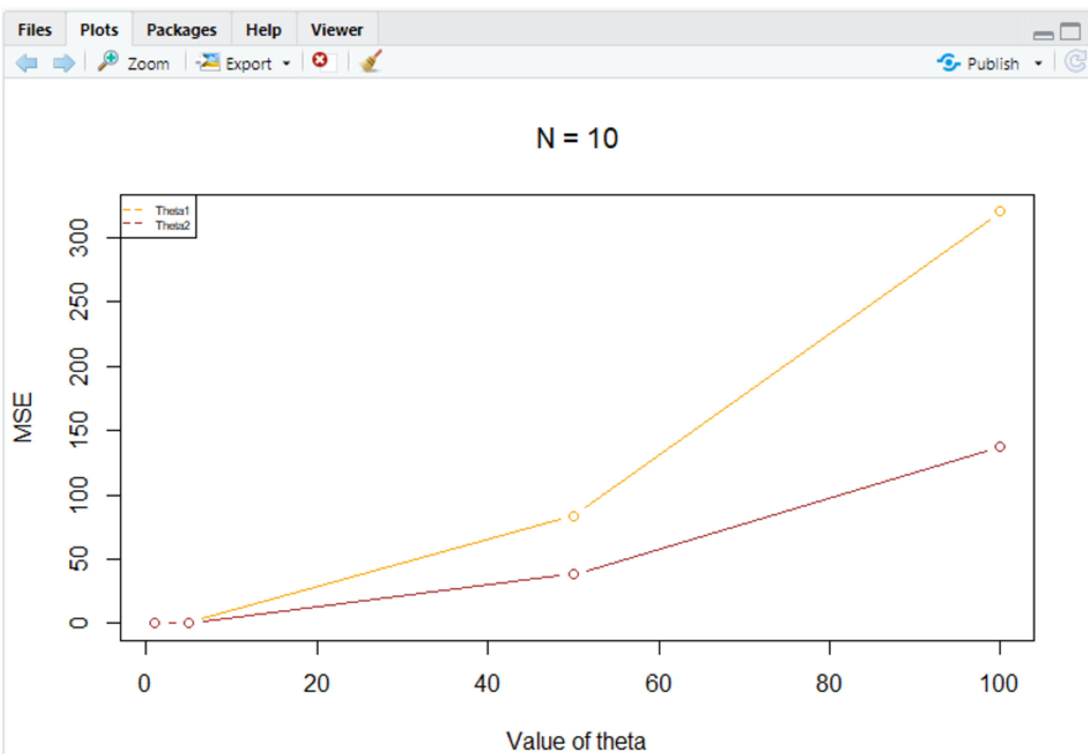
```
  MSE_Calc_theta2[counter] = calc[2]

  counter = counter + 1

 }


 plot(vals_of_n,MSE_cal_theta1,ylab = 'Mean squared Error',main=bquote(paste("Theta Value = ", .(itr1))), xlab = 'N value',

      type = 'b',col='orange')

 lines(vals_of_n,MSE_Calc_theta2,col='brown',type = 'b')

 legend("topleft",legend=c("Theta1","Theta2"),col=c('orange','brown'),cex = 0.5,lty=c(2,2),merge = TRUE)

}
```

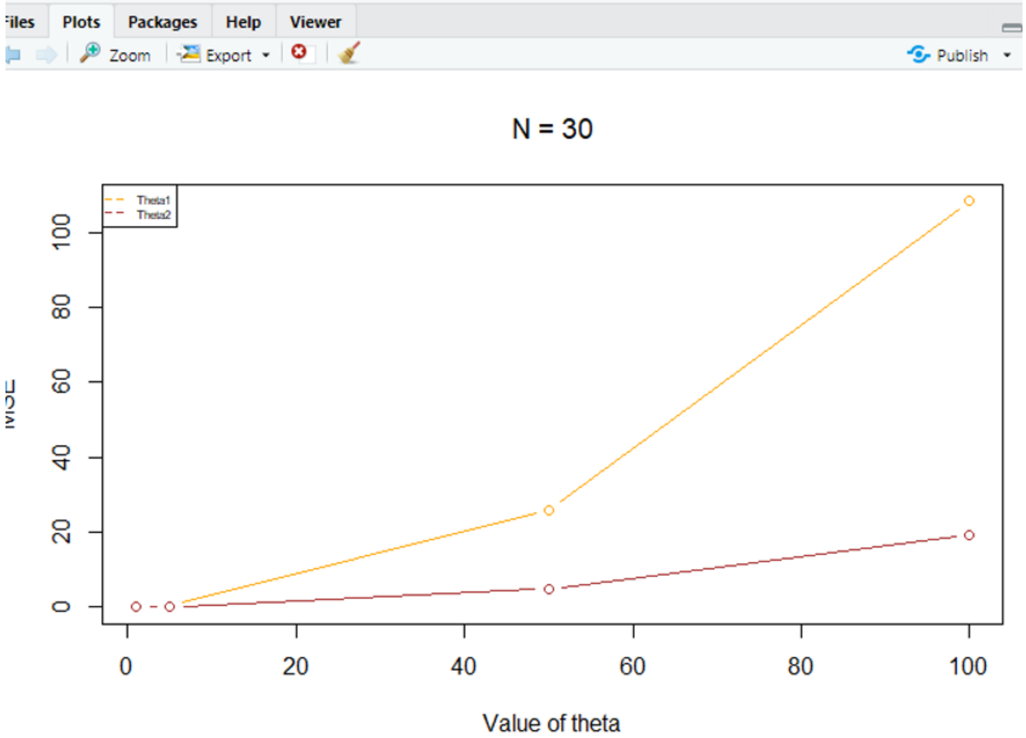



Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
N = 2
Theta1
Theta2
MSE
0
500
1000
1500
0
20
40
60
80
100
Value of theta

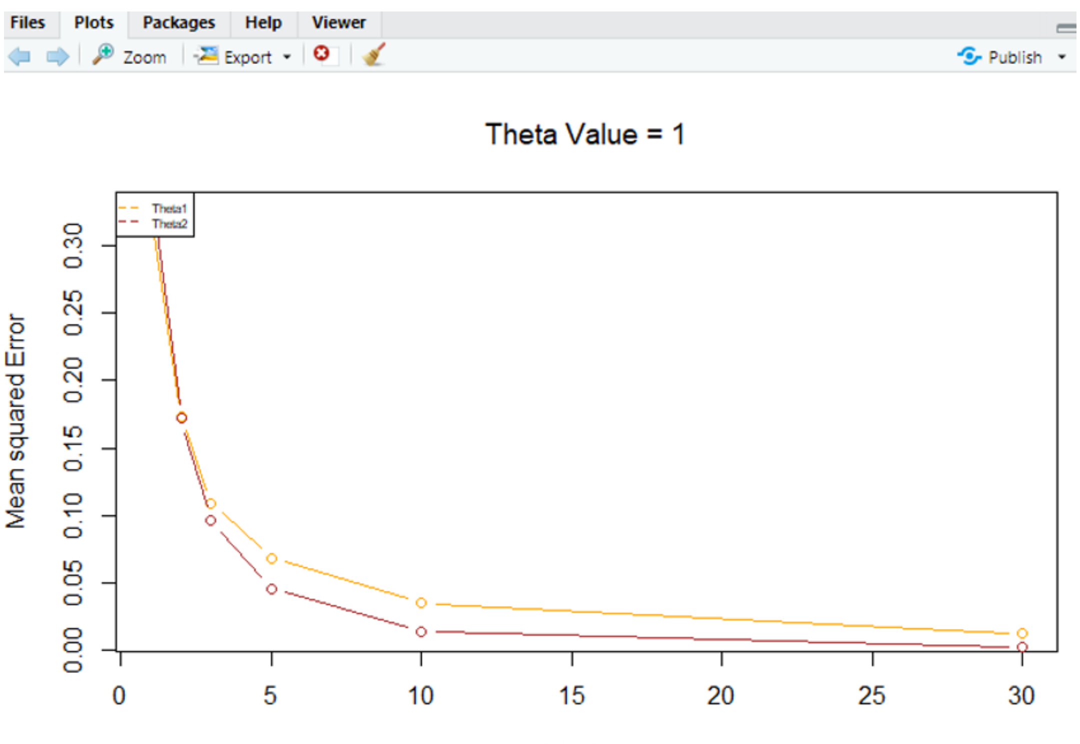

Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
N = 3
Theta1
Theta2
MSE
0
200
400
600
800
1000
0
20
40
60
80
100
Value of theta


Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
N = 5
Theta1
Theta2
MSE
0
100
200
300
400
500
600
0
20
40
60
80
100
Value of theta

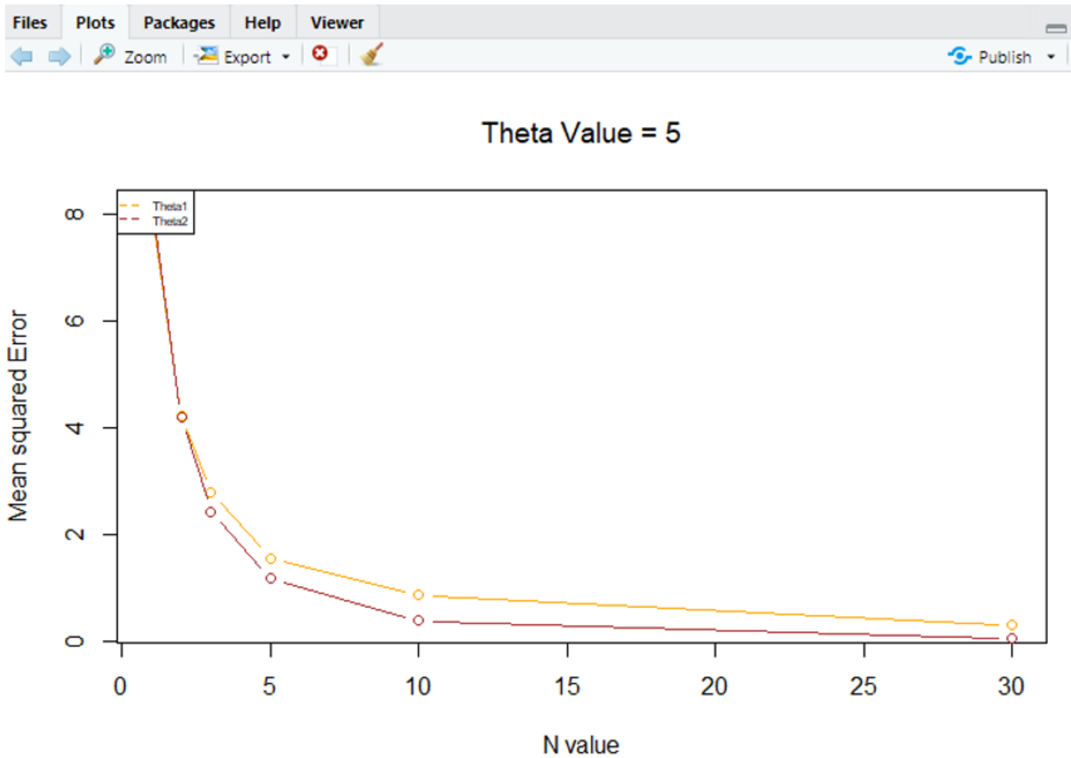

Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
N = 10
Theta1
Theta2
MSE
0
50
100
150
200
250
300
0
20
40
60
80
100
Value of theta



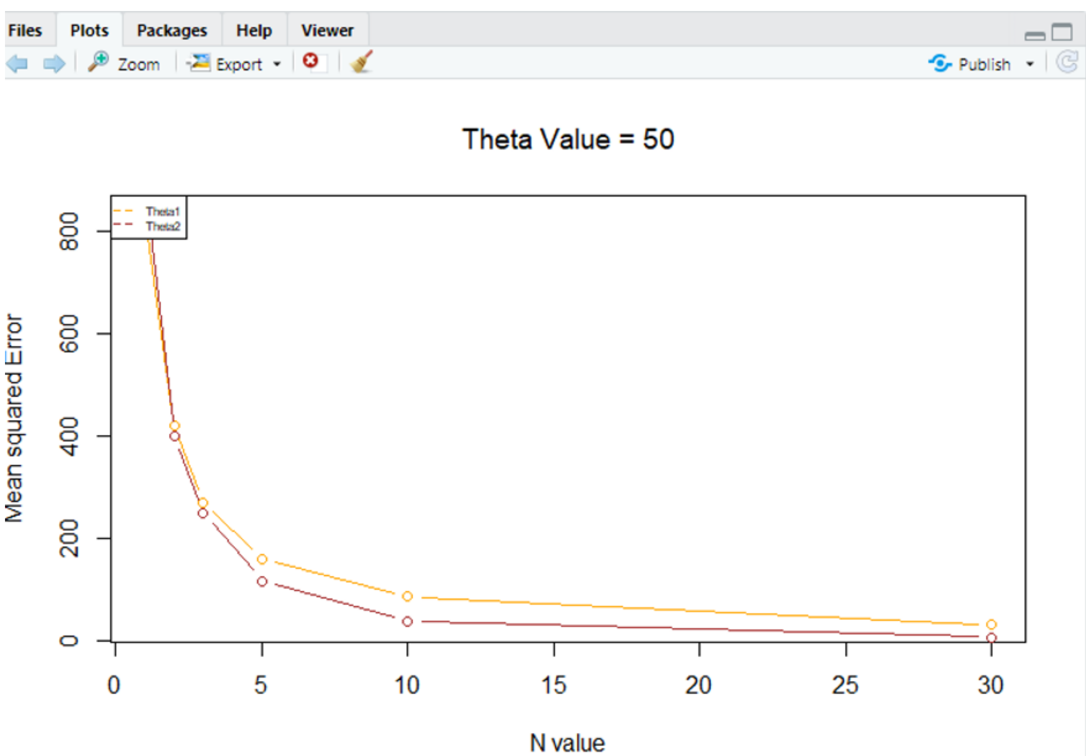
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
N = 30
Theta1
Theta2
0
20
40
60
80
100
Value of theta



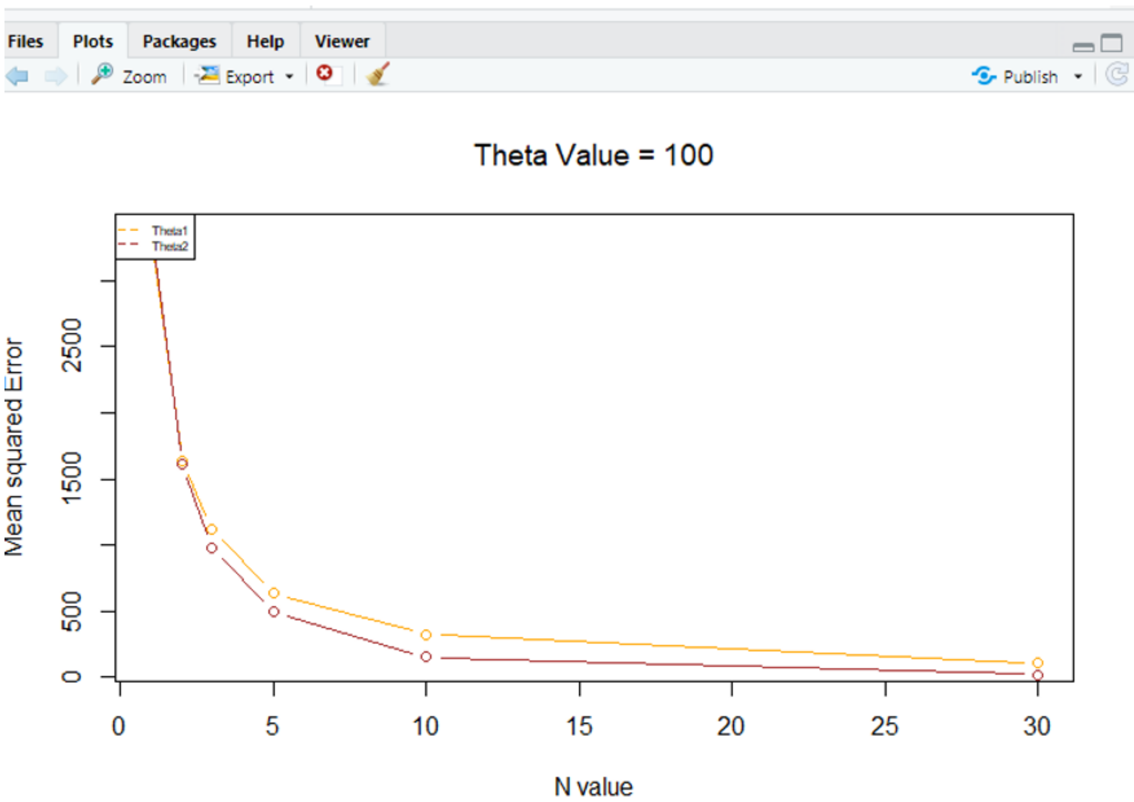
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Theta Value = 1
Theta1
Theta2
Mean squared Error
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0
5
10
15
20
25
30



Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Theta Value = 5
Theta1
Theta2
Mean squared Error
0
2
4
6
8
0
5
10
15
20
25
30
N value



Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Theta Value = 50
Theta1
Theta2
Mean squared Error
0
200
400
600
800
0
5
10
15
20
25
30
N value



1.d)

It can be inferred that as the sample value increases mean squared error goes on decreasing also MLE is better than Moments estimatoras it shows lower MSE, thus we infer MLE is better for bigger values n
#in the second case Theta values might differ but the resulting graphs are similar, thus inferring that estimator wouldn't depend on theta value
#Thus MLE is better than method of moments as sample size increases

2.a)

Question 2.

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

MLE of θ.

Soln: We denote MLE with $L(\theta)$ and in here, we have

$$L(\theta) = \prod_{i=1}^{n} \left(\frac{\theta}{x_i^{\theta+1}}\right)$$

Since, this is a product, taking logarithm on both sidee it becomes a summation & is strictly increasing. In probability theory, we take natural logarithm.

$$\log_e(L(\theta)) = \log_e\left(\prod_{i=1}^{n}\left(\frac{\theta}{x_i^{\theta+1}}\right)\right).$$

θ n time is θ^n & we can take it out of summation,

$$\log_e(L(\theta) = \log_e\left(\theta^n \times \prod_{i=1}^{n}\left(\frac{1}{x_i^{\theta+1}}\right)\right)$$

$$\boxed{\begin{aligned} \log_e ab &= \log_e a + \log_e b. \\ \log a^n &= n \log a. \end{aligned}} \rightarrow (1)$$

using (1) we have,

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

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

$$= n \log_e \theta - \theta \sum_{i=1}^{n} \log_e x_i - \sum_{i=1}^{n} \log_e x_i$$

On partially differentiating the above function, 2.b)

$$\frac{\partial (n \log_e \theta)}{\partial \theta} = \frac{n}{\theta} \quad , \quad \frac{\partial (\log_e \theta)}{\partial \theta} = \frac{1}{\theta} \quad ,$$

$$\frac{\partial \theta}{\partial \theta} \cdot \sum_{i=1}^{n} \log_e x_i = \sum_{i=1}^{n} \log_e x_i \left(\frac{\partial \theta}{\partial \theta} = 1 \right) \quad , \quad \frac{\partial \left(\sum_{i=1}^{n} \log_e x_i\right)}{\partial \theta} = 0$$

we have,

$$\log_e(L(\theta)) = \frac{n}{\theta} - \sum_{i=1}^{n} \log_e x_i$$

To get the MLE, we equate to 0.

$$\frac{n}{\theta} - \sum_{i=1}^{n} \log_e x_i = 0$$

$$\frac{n}{\theta} = \sum_{i=1}^{n} \log_e x_i$$

$$\boxed{\hat{\theta}_{MLE} = \frac{n}{\sum_{i=1}^{n} \log_e x_i}} \rightarrow (2)$$

b) The values can be inserted into the equation (2)

$$\hat{\theta}_{MLE} = \frac{n}{\sum_{i=1}^{n} \log_e x_i}$$

$x_1 = 21.72$, $x_2 = 14.65$,
$x_3 = 50.42$, $x_4 = 28.78$
$x_5 = 11.23$, $n = 5$.

$$\hat{\theta}_{MLE} = \frac{5}{\log_e(21.72) + \log_e(14.65) + \log_e(50.42) + \log_e(28.78) + \log_e(11.23)}$$

$$\hat{\theta}_{MLE} = \frac{5}{\log_e(21.72 \times 14.65 \times 50.42 \times 28.78 \times 11.23)}$$

$$\hat{\theta}_{MLE} = \frac{5}{\log_e(5185263.523)}$$

$$\hat{\theta}_{MLE} = \frac{5}{15.461}$$

$$\boxed{\hat{\theta}_{MLE} = 0.323}$$

2.c)

```
#Add the values to a vector
max_log_func <- c(21.72,14.65,50.42,28.78,11.23)

#writing a function to get the likelihood parameter
log_likelihood <- function(par,data_pt){
 res = length(data_pt)*log(par)-(par+1)*sum(log(data_pt))
 return(-res)
}

#using optim function and giving hessian as true
max_likely <-optim(par = 0.5,data_pt=max_log_func,
fn=log_likelihood,method="L-BFGS-B",hessian = TRUE,lower = 0.01)
max_likely
```

```
> max_log_func <- c(21.72,14.65,50.42,28.78,11.23)
>
> #writing a function to get the likelihood parameter
> log_likelihood <- function(par,data_pt){
+   res = length(data_pt)*log(par)-(par+1)*sum(log(data_pt))
+   return(-res)
+ }
>
> #using optim function and giving hessian as true
> max_likely <-optim(par = 0.5,data_pt=max_log_func, fn=log_likelihood,method="L-BFGS-B",hessian = TRUE,lower = 0.01)
> max_likely
$par
[1] 0.3233885

$value
[1] 26.10585

$counts
function gradient
       6        6

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"

$hessian
         [,1]
[1,] 47.81116
```

#result

#Yes the answers match

2.d)

#We need to get the hessian matrix to calculate this estimate

err<- sqrt(diag(solve(max_likely$hessian)))

err

alp_val <- 0.05

confidence interval is mean +/- z-score multiplied by error, we use qnorm function to find it

Confidence_Int = max_likely$par + c(-1, 1)* qnorm(1-(alp_val/2)) * err

Confidence_Int

```
> #We need to get the hessian matrix to calculate this estimate
> err<- sqrt( diag(solve(max_likely$hessian)))
> err
[1] 0.1446223
>
>
> alp_val <- 0.05
> # confidence interval is mean +/- z-score multiplied by  error, we use qnorm function to find it
> Confidence_Int = max_likely$par + c(-1, 1)* qnorm(1-(alp_val/2)) * err
> Confidence_Int
[1] 0.0399339 0.6068430
> |
```

#infer-> The true estimate lies within the given confidence interval i,e out of 100 trials 95 percent of the trials will be there

#in the given interval