**Statistical Methods for Data Science**

Project 3

By: Chirag Shahi cxs180005

Krishnan Vijayaraghavan kxv190006

Contribution:

Chirag: Question 1

Krishnan: Question 2

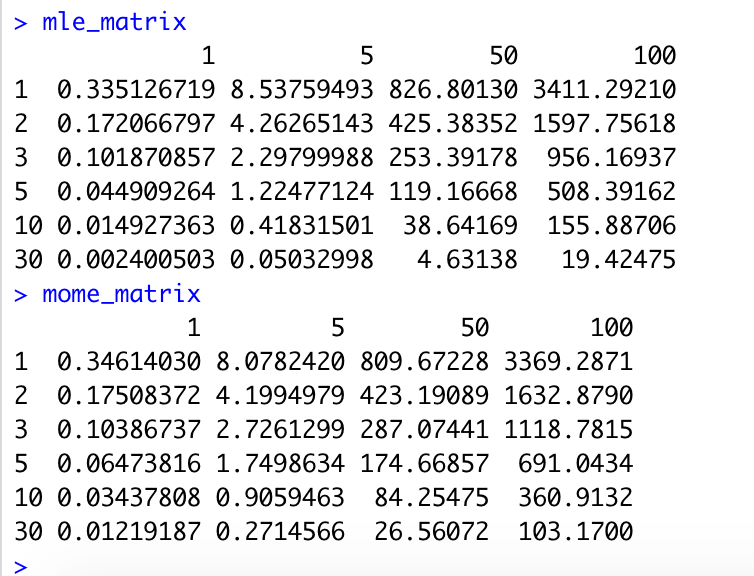
**SECTION 1**

1.a)

Monte Carlo method relies on repeated random sampling to obtain results. We are given a Uniform distribution so we use ‘runif’ in R to generate random samples. Squared error of an estimator is the square of the difference between the estimated value using the estimator and the original parameter value. So we need to find the estimated value of both the given estimators which is maximum of the sample for MLE and twice the mean of the sample for MOME. We run iterations and store the value of the squared error each time. Now to find the mean squared error, we find the average of the squared errors.

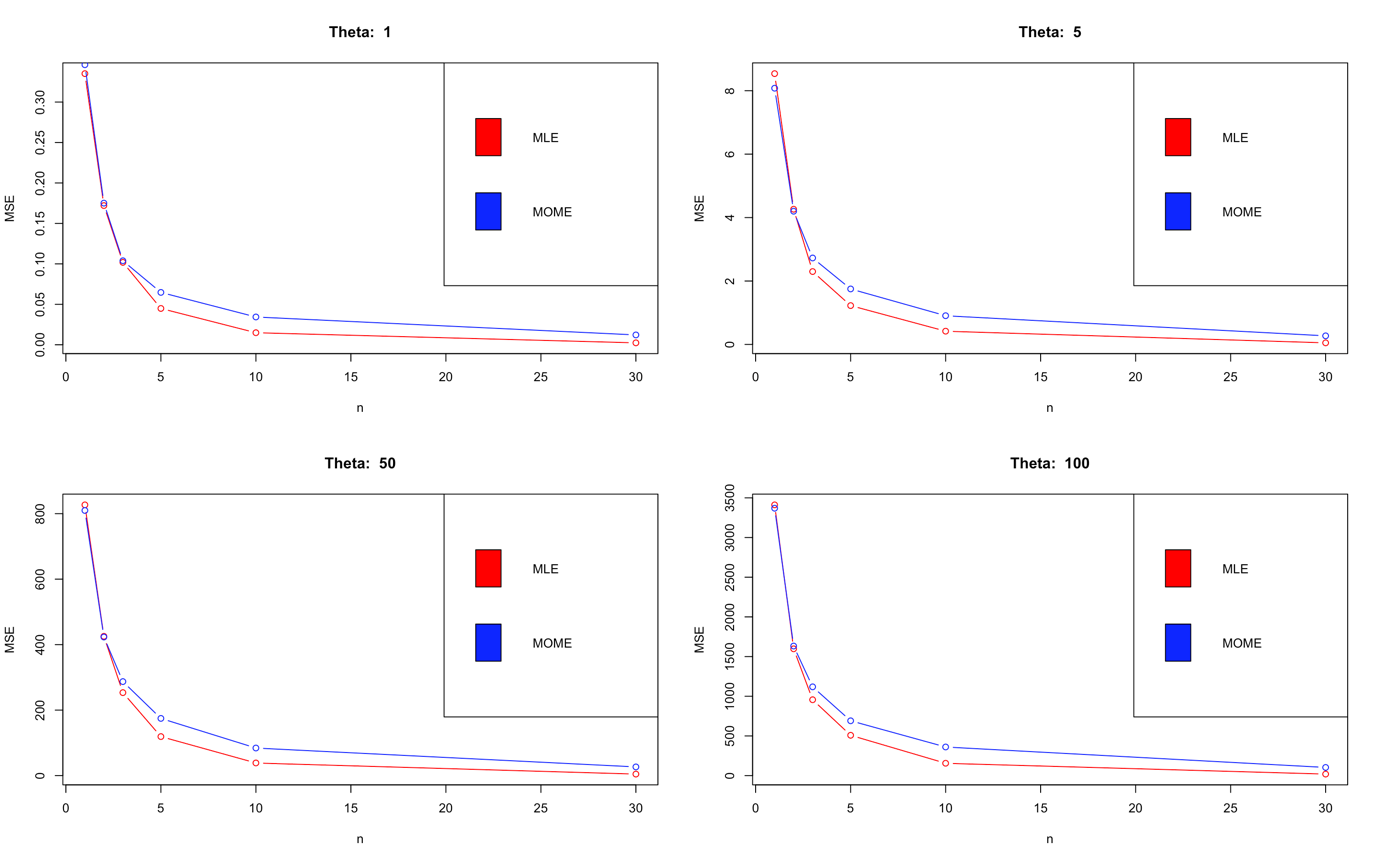
1.b)

Below is the matrix representation of the MSE of MLE followed by MSE of MOME of different combinations of ‘n’ & ‘theta’ where the rows represent the different values of ‘n’ and columns the different values of ‘theta’ after 1000 iterations:



1.c)

Below are 4 plots for 4 different values of ‘theta’ comparing the MSE for MLE & MOME for different values of n. MLE is represented by the red line and MOME by the blue line.



1.d)

In all the 4 plots, the MSE for MLE is lower than that of MOME for all values of n >= 3. For values of n > 0 and n < 3, the difference between the MSE’s is negligibly low and can be treated as equal. Thus, MLE is a better estimator overall and the answer does not depend upon ‘n’. It does not depend upon ‘theta’ either as increasing ‘theta’ leads to an increase in MSE as well for both the estimators.

2.a) sum{log(theta) -(theta+1)\*log(x)}

theta=n/sum(log(x))

2.b) 0.3233874

2.c) 0.3233387

The two values are almost similar.

2.d) Standard error 0.1446217

Confidence Interval: [25.18,25.34]

The mean of the confidence interval 25.36 lies in the confidence interval.

So, it is a good approximation.

**SECTION 2**

1)

n\_vals = c(1,2,3,5,10,30)

theta\_vals = c(1,5,50,100)

mse <- function(n, theta) {

estimates <- function(n, theta) {

# random draws

x <- runif(n, min = 0, max = theta)

# mle = max of sample

mle <- max(x)

# mome = twice of mean of sample

mome <- 2 \* mean(x)

return (c(mle = mle, mome = mome))

}

# replicate 1000 times

estimatesAfterReplications <- replicate(1000, estimates(n, theta))

# return mse values for mle, mome respectively

return (rowMeans((estimatesAfterReplications - theta) ^ 2))

}

# create MSE matrix for MLE

mle\_matrix = matrix(nrow = length(n\_vals), ncol = length(theta\_vals))

rownames(mle\_matrix) <- n\_vals

colnames(mle\_matrix) <- theta\_vals

# create MSE matrix for MOME

mome\_matrix = matrix(nrow = length(n\_vals), ncol = length(theta\_vals))

rownames(mome\_matrix) <- n\_vals

colnames(mome\_matrix) <- theta\_vals

# fill values in the matrices

for (i in 1:length(n\_vals)) {

for (j in 1:length(theta\_vals)) {

value <- mse(n\_vals[i], theta\_vals[j])

mle\_matrix[i, j] <- value["mle"]

mome\_matrix[i, j] <- value["mome"]

}

}

# print matrices

mle\_matrix

mome\_matrix

# dividing plots into 2X2 array of subplots

par(mfrow = c(2, 2))

# subplots for each value of theta

for (i in 1:length(theta\_vals)) {

plot(n\_vals, mle\_matrix[,i],

# plotting lines and points both

type = "b",

col = "red",

ylab = "MSE",

xlab = "n",

main = paste("Theta: ", theta\_vals[i]),

)

lines(n\_vals, mome\_matrix[,i],

col = "blue",

type = "b"

)

legend("topright", c("MLE", "MOME"),

fill = c("red", "blue"))

}

2)

a) # Log likelihood function

log.lik=function(theta,x){

n=nrow(x)

logl= -n\*log(theta)-(theta+1)\*log(sum(x))

return(-logl)}

# Expression of maximum likelihood estimator

n\*log(theta)-(theta+1)\*log(sum(x))=0

# Equating to zero to find maximum likelihood estimator

theta=n/sum(log(x))

b) n=5

x=c (21.72, 14.65,50.42,28.78,11.23)

mle=n/sum(log(x))

print(mle)

# Finding the value of theta using given sample

c) fn=function(par,dat){

result=sum(log(par/dat\*\*(par+1)))

return(-result)}

ml.est=optim(par=1,fn=fn,dat=x,method=”BGFS”,hessian=TRUE)

ml.est

# Maximizing log likelihood using optim function. The function takes the

arguments parameter, log-likelihood, data among many others

d) stderr= sqrt(diag(solve(ml.est$hessian)))

print(stderror)

mu= mean(x)

print(mu)

alpha= 0.05

n= 5

conf.int=function(mu, stderr,n , alpha){

ci=round(mu + c(-1,1)\*qt(1-(alpha/2), df = (n-1)) \* stderr/sqrt(n), 3)

return(ci)

}

# Computing the confidence intervals

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |