Project Report

Project – 3

Course Name - Statistical Methods for Data Science

Name of group members

Gnaneswar Gandu (Net id – GXG170000)

Lakshmi Sindhuri Pallapothu (Net id – LXP170004)

Contribution:

- Solved Exercise 1&2 together.
- R code for Question 1 was done by Gnaneswar and for Question 2 was done by Sindhuri.
- Executed Histogram related views for several draws.
- Debugged Errors during R commands execution.

Section -1

1. **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, $\hat{\theta}_1 = X(n)$, where X(n) is the maximum of the sample, and the method of moments estimator, $\hat{\theta}_2 = 2X$, 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 $\hat{\theta}$ of a parameter θ is defined as $E\{(\hat{\theta} - \theta)^2\}$. 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.

   First a population parameter is set.
   Next the sample values are simulated out of which we calculate the estimator value then the mean squared error is nothing but the estimated value of the squared
   difference between the estimator and the parameter.

   (b) For a given combination of (n, θ), compute the mean squared errors of both $\hat{\theta}_1$ and $\hat{\theta}_2$ using Monte Carlo simulation with N = 1000 replications. Be sure to compute both estimates from the same data.

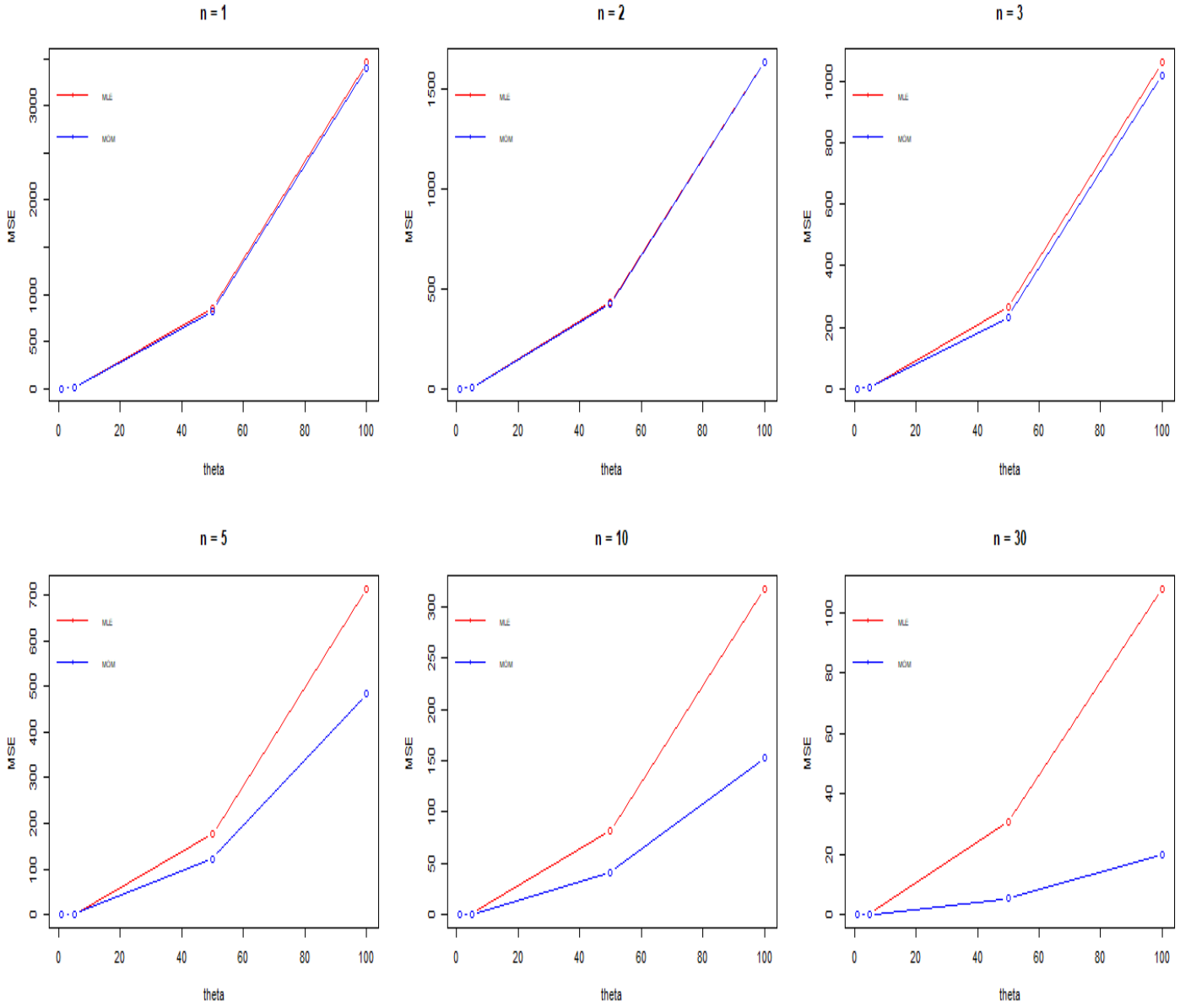
   (Refer to Section 2 for R code)
   We wrote a small function (called mom_mle_func(n, θ) that takes a sample
   from uniform distribution and calculates the MOM and MLE for that sample
   based on the given information and returns them as an array of two values. We also

created a new function called mse_estimator_func(n, θ) which will call the previous function 1000 times and then calculate the mean squared errors using the formula $E\{(\theta\hat{} - \theta)\}$ and
return the mean squared errors for both the estimators as an array
Let us take the first combination of n and Θ as (1, 1)
The mean squared errors are:

$\mathrm{MSE}_{\theta 1\hat{}}$ = 0.3589130
$\mathrm{MSE}_{\theta 2\hat{}}$ = 0.3380401

(c) Repeat (b) for the remaining combinations of (n, Θ). Summarize your results
graphically.
(Refer Section 2 for R code)
We use the function par to change the layout of the plot output to accommodate more
than one graph
In our case, we are using par(mfrow = c(2,2))
Here mfrow is used to position the subsequent graphs as the given parameters
suggest. So in this case the subsequent four graphs are put in a 2x2 matrix.
The graphs we get are:

GRAPH 1: Mean squared errors of MLE and MOM, Θ with fixed n



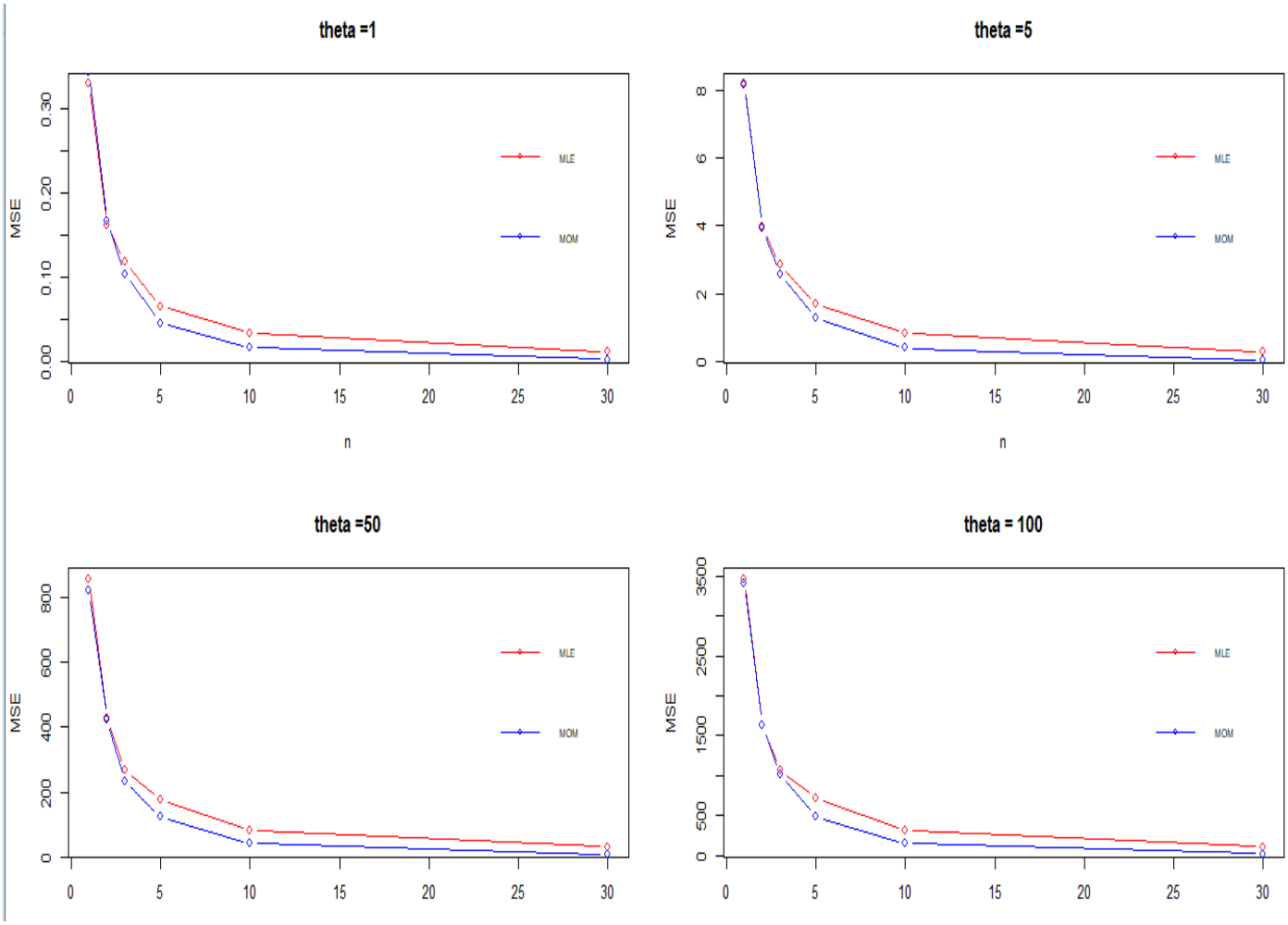
Observations:

- From above results we can observe that, the MSE's of both the estimators decrease as
n increases, which means that as estimators become more accurate as n increases.
- They converge to the true parameter value in the given limit where MSE can be 0.
- It can also be seen that for n=1, MSEs of both estimators are almost same and when

value of n is increased, like n>=3, MSE for MLE (theta 1) lies below that of MOME (theta
2).

• So, we can say that n=1, the two estimators hold good importance, but n>=3, the MLE is
better when compared.

GRAPH 2: Mean squared errors of MLE and MOM, n with fixed Θ



Observations:

- From above results we can observe that, the MSE's of both the estimators is same as
Θ increases, which means that as estimators won't change with Θ.

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

We can see from the observations of GRAPH 2 from (c) that no matter what value of we fixed, we are getting the similar graph. This can also be interpreted as that the estimator doesn't depend on the value of Θ. Now looking carefully at the GRAPH 1 where we plotted the values of mean squared errors varying with fixed n, it is very evident that we can use Method of Moments estimator for small values of n (=1, 2, 3). But as the n (=5, 10, 30) value
increases, the Maximum Likelihood Estimator is better. Hence for larger n, MLE is
better or as n increases the Maximum Likelihood Estimator is preferred. The MLE is
preferred as its mean squared error is less for the same n when compared with MOM.

Section - 2

R code for Question 1

(b) #calculating the mean squared errors

```
#creating a function to return MOM and MLE for the same sample
>mom_mle_func = function(n, theta) {
  sample = runif(n, min = 0, max = theta)
  mom_estimator = 2*mean(sample)
  mle_estimator = max(sample)
  return(c(mom_estimator, mle_estimator))
}

#creating a function to calculate and return the mean squared errors of mle and mom
```

for 1000 replications

```
>mse_estimator_func = function(n, theta) {
  estimates = replicate(1000, mom_mle_func(n, theta))
  estimates = (estimates-theta)^2
  estimates.mom_estimator = estimates[c(TRUE, FALSE)]
  estimates.mle_estimator = estimates[c(FALSE, TRUE)]
  return (c(mean(estimates.mom_estimator),
mean(estimates.mle_estimator)))
}
```

#finding mean squared errors for all one combination of n and Θ (1,1)

```
>mse_est_1_1 = mse_estimator_func(1,1)
>mse_est_1_1
  [1] 0.3589130 0.3380401
```

(c) #drawing graphs of mean squared errors

#finding mean squared errors for all combinations of n and Θ

```
> mse_est_1_5 = mse_estimator_func(1,5)
> mse_est_1_50 = mse_estimator_func(1,50)
> mse_est_1_100 = mse_estimator_func(1,100)
> mse_est_2_1 = mse_estimator_func(2,1)
> mse_est_2_5 = mse_estimator_func(2,5)
> mse_est_2_50 = mse_estimator_func(2,50)
> mse_est_2_100 = mse_estimator_func(2,100)
> mse_est_3_1 = mse_estimator_func(3,1)
> mse_est_3_5 = mse_estimator_func(3,5)
> mse_est_3_50 = mse_estimator_func(3,50)
> mse_est_3_100 = mse_estimator_func(3,100)
> mse_est_5_1 = mse_estimator_func(5,1)
> mse_est_5_5 = mse_estimator_func(5,5)
> mse_est_5_50 = mse_estimator_func(5,50)
> mse_est_5_100 = mse_estimator_func(5,100)
> mse_est_10_1 = mse_estimator_func(10,1)
> mse_est_10_5 = mse_estimator_func(10,5)
```

```
> mse_est_10_50 = mse_estimator_func(10,50)
> mse_est_10_100 = mse_estimator_func(10,100)
> mse_est_30_1 = mse_estimator_func(30,1)
> mse_est_30_5 = mse_estimator_func(30,5)
> mse_est_30_50 = mse_estimator_func(30,50)
> mse_est_30_100 = mse_estimator_func(30,100)

#drawing graphs with fixed n value and varying Θ

>par(mfrow=c(2,3))
>plot(c(1,5,50,100), c(mse_est_1_1[1],mse_est_1_5[1], mse_est_1_50[1],
mse_est_1_100[1]), type="b", xlab = 'theta', ylab = 'MSE', col = 'red', main =
"n = 1")
>lines(c(1,5,50,100), c(mse_est_1_1[2],mse_est_1_5[2], mse_est_1_50[2],
mse_est_1_100[2]), type="b", col = 'blue')
>legend("topleft",legend=c("MLE","MOM"),col=c('red','blue'),text.col=c('bla
ck','black'),lty=1,pch=1,inset=0.01,ncol=1,cex=0.6,bty='n')
>plot(c(1,5,50,100), c(mse_est_2_1[1],mse_est_2_5[1], mse_est_2_50[1],
mse_est_2_100[1]),type="b", xlab = 'theta', ylab = 'MSE', col = 'red', main =
"n = 2")
>lines(c(1,5,50,100), c(mse_est_2_1[2],mse_est_2_5[2], mse_est_2_50[2],
mse_est_2_100[2]),type="b", col = 'blue')
>legend("topleft", 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(c(1,5,50,100), c(mse_est_3_1[1],mse_est_3_5[1], mse_est_3_50[1],
mse_est_3_100[1]),type="b", xlab = 'theta', ylab = 'MSE', col = 'red', main =
"n = 3")
>lines(c(1,5,50,100), c(mse_est_3_1[2],mse_est_3_5[2], mse_est_3_50[2],
mse_est_3_100[2]),type="b", col = 'blue')
>legend("topleft", 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(c(1,5,50,100), c(mse_est_5_1[1],mse_est_5_5[1], mse_est_5_50[1],
mse_est_5_100[1]),type="b", xlab = 'theta', ylab = 'MSE', col = 'red', main =
"n = 5")
>lines(c(1,5,50,100), c(mse_est_5_1[2],mse_est_5_5[2], mse_est_5_50[2],
mse_est_5_100[2]),type="b", col = 'blue')
```

```
>legend("topleft", 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(c(1,5,50,100), c(mse_est_10_1[1],mse_est_10_5[1],
mse_est_10_50[1], mse_est_10_100[1]),type="b", xlab = 'theta', ylab =
'MSE', col = 'red', main = "n = 10")
>lines(c(1,5,50,100), c(mse_est_10_1[2],mse_est_10_5[2],
mse_est_10_50[2], mse_est_10_100[2]),type="b", col = 'blue')
>legend("topleft", 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(c(1,5,50,100), c(mse_est_30_1[1],mse_est_30_5[1],
mse_est_30_50[1], mse_est_30_100[1]),type="b", xlab = 'theta', ylab =
'MSE', col = 'red', main = "n = 30")
>lines(c(1,5,50,100), c(mse_est_30_1[2],mse_est_30_5[2],
mse_est_30_50[2], mse_est_30_100[2]),type="b", col = 'blue')
>legend("topleft", 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')

#drawing graphs with fixed Θ value and varying n

>par(mfrow=c(2,2))
>plot(c(1,2,3,5,10,30), c(mse_est_1_1[1],mse_est_2_1[1], mse_est_3_1[1],
mse_est_5_1[1],mse_est_10_1[1], mse_est_30_1[1]), type="b", ylab =
'MSE', xlab = 'n', col = 'red', main = "theta =1")
>lines(c(1,2,3,5,10,30), c(mse_est_1_1[2],mse_est_2_1[2], mse_est_3_1[2],
mse_est_5_1[2],mse_est_10_1[2], mse_est_30_1[2]), type="b", col =
'blue')
>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(c(1,2,3,5,10,30), c(mse_est_1_5[1],mse_est_2_5[1], mse_est_3_5[1],
mse_est_5_5[1],mse_est_10_5[1], mse_est_30_5[1]), type="b", ylab =
'MSE', xlab = 'n', col = 'red', main = "theta =5")
>lines(c(1,2,3,5,10,30), c(mse_est_1_5[2],mse_est_2_5[2], mse_est_3_5[2],
mse_est_5_5[2],mse_est_10_5[2], mse_est_30_5[2]), type="b", col =
'blue')
>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(c(1,2,3,5,10,30), c(mse_est_1_50[1],mse_est_2_50[1],
mse_est_3_50[1], mse_est_5_50[1],mse_est_10_50[1],
mse_est_30_50[1]), type="b", ylab = 'MSE', xlab = 'n', col = 'red', main =
"theta =50")
>lines(c(1,2,3,5,10,30), c(mse_est_1_50[2],mse_est_2_50[2],
mse_est_3_50[2], mse_est_5_50[2],mse_est_10_50[2],
mse_est_30_50[2]), type="b", col = 'blue')
>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(c(1,2,3,5,10,30), c(mse_est_1_100[1],mse_est_2_100[1],
mse_est_3_100[1],mse_est_5_100[1], mse_est_10_100[1],
mse_est_30_100[1]), type="b", ylab = 'MSE', xlab = 'n', col ='red', main =
"theta = 100")
>lines(c(1,2,3,5,10,30), c(mse_est_1_100[2],mse_est_2_100[2],
mse_est_3_100[2],mse_est_5_100[2], mse_est_10_100[2],
mse_est_30_100[2]), type="b", col = 'blue')
>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')
```

R code for Question 2

(c) #R code for minimizing the function

```
 #creating a function to return the negative value of the derived function
>neg_loglikelyhood_func = function(par, dat) {
          result = length(dat) * log(par) - (par + 1) * sum(log(dat))
          return(-result)
        }


#initialising an aray with the given data

>x = c(21.42, 14.65, 50.42, 28.78, 11.23)

#executing the optim function for minimizing the derived function negative
value
>mle <- optim(par=0.926, fn=neg_loglikelyhood_func, method="L-BFGS-
B",hessian=TRUE, lower=0.01, dat=x)
```

```
>mle

 $par
 [1] 0.3236796

$value
[1] 26.08744

$counts
function gradient
      10       10

$convergence
[1] 0

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

$hessian
          [,1]
[1,] 47.72518
```

(d) #R code for finding the confidence interval

```
 #finding the standard error
> se = (1/mle$hessian)^0.5
> se
       [,1]
 [1,] 0.1447525

#finding the confidence interval


>  mle$par + c(-1,1)*se*qnorm(0.975)
[1] 0.03996985 0.60738940
```