Mini Project #3
Tanushri Singh, Nikhil Pareek
Contribution of each group member:
Both worked together to finish the two questions. Worked together to understand the problem sets and and then wrote the scripts for problems one and two. Nikhil worked to check the accuracy of the script and Tanu worked to Document both questions and report all the findings. Both partners worked efficiently to complete the project requirements!

**NOTE: ALL CODES ARE ATTACHED IN SECTION 2!

<u>Section #1</u>

Question 1

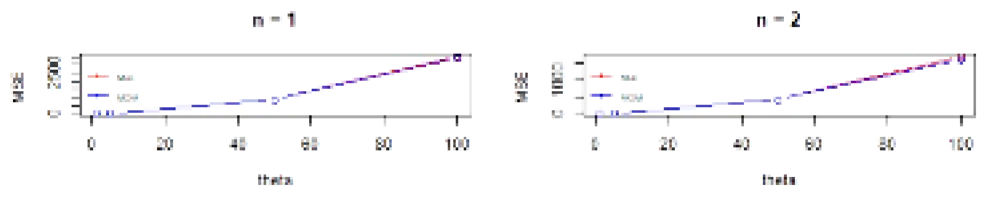
(a) Mean squared error can be calculated by first setting a population parameter then simulating the sample values which will allow for the calculation of the estimator value. Now the mean squared error is nothing but the estimated value of the difference squared amongst the estimator and parameter.

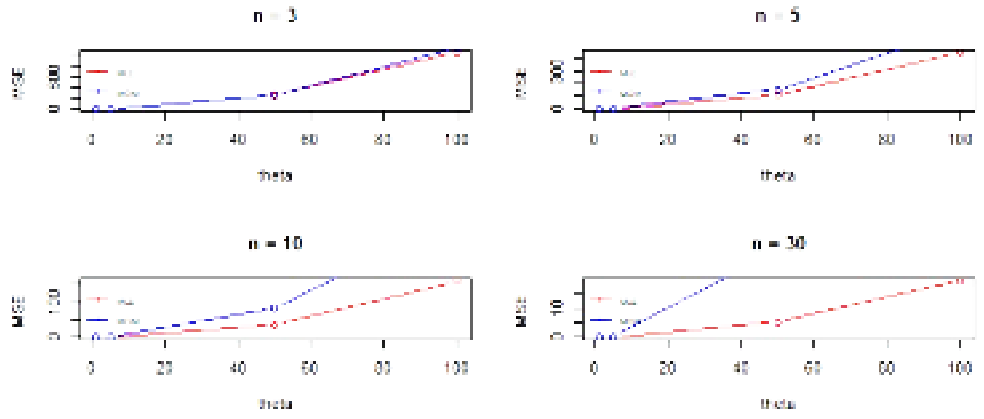
(b) R code is included in section 2!
The function mleandmom(n, θ)) simulates samples from a uniform distribution and then calculates the MLE as well as MOM for the given sample, then it will return an array of two values. The function msees(n, θ)) will call the mleandmom() function 1 thousand times and it will calculate the mean squared error using the formula E{(θ - θ)^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 θ as (1, 1) so we get the mean squared errors as:

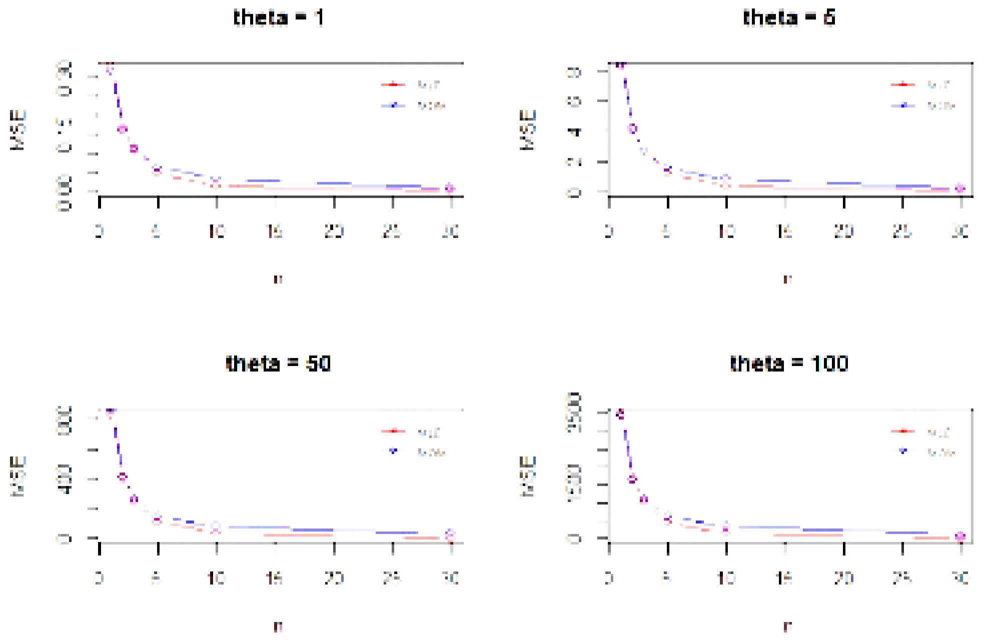
$MSE(\theta_1)$ = 0.3252497 $MSE(\theta_2)$ = 0.3133334

(c) R code is included in section 2!
Function par() is used to change the layout of the plot output. This is done in order to accommodate more than one graph, the syntax for this is par(mfrow = c(2,2)). In which mfrow is used to position the subsequent graphs. Hence, the subsequent four graphs are placed in matrix of size 2x2.





Graph 1: Mean squared errors of MLE and MOM, θ with fixed n



Graph 2: Mean squared errors of MLE and MOM, n with fixed θ

(d) It can be seen from the second graph that no matter what value of θ get fixed, the resulting graphs are extremely similar. So, it can be interpreted that the estimator wouldn't depend on the value of θ. Graph 1 depicts the plotted values of mean squared errors varying with θ 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. 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.

Question 2:

(a) Take likelihood function as $L(\theta) = \prod_{i=1}^{n}(\frac{\theta}{x_i^{\theta+1}})$

Now take the log of both sides and the resulting answer would be:

$$\begin{aligned}\log(L(\theta)) &= log(\prod_{i=1}^{n}(\frac{\theta}{x_i^{\theta+1}}))\\ &= log(\theta^n \times \prod_{i=1}^{n}\frac{1}{x_i^{\theta+1}})\\ &= nlog\theta \ + \sum_{i=1}^{n} log\left(x_i^{-\theta-1}\right)\\ &= nlog\theta - (\theta+1)\sum_{i=1}^{n} logx_i\\ &= nlog\theta - \theta\sum_{i=1}^{n} logx_i - \sum_{i=1}^{n} logx_i\end{aligned}$$

When the equation is partially differentiated it results in:

$$= \frac{n}{\theta} - \sum_{i=1}^{n} logx_i$$

Once the equation is equated to 0, the resulting answer is:

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

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

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

(b) Values need to be plugged into the equation. Resulting answer is:

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

$$\hat{\theta}_{MLE} = \frac{5}{\log(21.52 \times 14.65 \times 50.42 \times 28.78 \times 11.23)}$$

$$\hat{\theta}_{MLE} = \frac{5}{\log(5137517.08)}$$

$$\hat{\theta}_{MLE} = \frac{5}{15.45}$$

$$\hat{\theta}_{MLE} = 0.3236$$

(c) R Code can be found in Section 2!

In R, there is a provision to minimize any function that is given. In this scenario we require maximization to occur. Hence the negative of the function must be maximized. Which is why the function gets negated and then minimized in R.

The function that gets used in R has the following Syntax: optim(par, fn, ...)

The Function performs minimization on the function given

Here: par = initial values for the parameters to be optimized over
fn = the function being minimized

As depicted through the results of the R code, the estimate is 0.3236.

(d)

The formula for standard error is given as:

$SE(\hat{\theta}) \approx \sqrt{\hat{I}^{-1}}$

Where $\hat{I}$ is the hessian function

(Refer to Section 2 for R code)

From R code we get the value of SE as 0.1447525

Given $\mathbf{1} - \alpha = 0.95$

$\alpha = 1 - 0.95$

$\alpha = 0.05$

And $\frac{\alpha}{2} = \frac{0.05}{2} = 0.025$

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

The confidence interval formula is given as: $\hat{\theta} \perp Z_{\frac{\alpha}{2}} \times SE(\hat{\theta})$

In R we use qnorm function to get the $Z_{\frac{\alpha}{2}}$ value

From R we get our confidence interval as

(0.03996985, 0.60738900)

So, we know that out of the 100 trials to get the population estimated, the true estimate value lies in the within the interval 95% of the times.

<u>Section # 2</u>

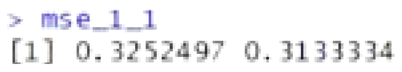
R code for Question 1

1b) #calculating the mean squared errors

```
#create function to return MLE/MOM for the same sample
> mleandandmom <- function(n, theta) {

    +   sample = runif(n, min = 0, max = theta)
    +   momes = 2*mean(sample)
    +   mlees = max(sample)
    +   return(c(mlees, momes))

+}

#create function to calculate and return mean squared errors of MLE/MOM for 1000 replications
> msees = function(n, theta) {

    +   estimates = replicate(1000, mleandandmom(n,theta))
    +   estimates = (estimates - theta)^2
    +   estimates.momes = estimates[c(TRUE,FALSE)]
    +   estimates.mlees = estimates[c(FALSE,TRUE)]
    +   return(c(mean(estimates.mlees), mean(estimates.momes)))

+}
```

#find mean squared errors for all combinations of n and θ (1,1)

```
> mse_1_1 = msees(1,1)
```



```
> mse_1_1
[1] 0.3252497 0.3133334
```

1c) #draw graphs of mean squared errors

```
#find mean squared errors for all combinations of n and θ > mse_1_5 = msees(1,5)
> mse_1_50 = msees(1,50)
> mse_1_100 = msees(1,100)
> mse_2_1 = msees(2,1)
> mse_2_5 = msees(2,5)
> mse_2_50 = msees(2,50)
> mse_2_100 = msees(2,100)
> mse_3_1 = msees(3,1)
> mse_3_5 = msees(3,5)
> mse_3_50 = msees(3,50)
> mse_3_100 = msees(3,100)
> mse_5_1 = msees(5,1)
> mse_5_5 = msees(5,5)
> mse_5_50 = msees(5,50)
> mse_5_100 = msees(5,100)
> mse_10_1 = msees(10,1)
> mse_10_5 = msees(10,5)
> mse_10_50 = msees(10,50)
> mse_10_100 = msees(10,100)
> mse_30_1 = msees(30,1)
> mse_30_5 = msees(30,5)
> mse_30_50 = msees(30,50)
> mse_30_100 = msees(30,100)
```

```
# draw graphs with fixed n value and varying θ
> par(mfrow=c(3,2))
> plot(c(1,5,50,100), c(mse_1_1[1],mse_1_5[1], mse_1_50[1], mse_1_100[1]), type="b",
xlab = 'theta', ylab = 'MSE', col = 'red', main = "n = 1")
> lines(c(1,5,50,100), c(mse_1_1[2],mse_1_5[2], mse_1_50[2], mse_1_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_2_1[1],mse_2_5[1], mse_2_50[1], mse_2_100[1]), type="b",
xlab = 'theta', ylab = 'MSE', col = 'red', main = "n = 2")
> lines(c(1,5,50,100), c(mse_2_1[2],mse_2_5[2], mse_2_50[2], mse_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_3_1[1],mse_3_5[1], mse_3_50[1], mse_3_100[1]), type="b",
xlab = 'theta', ylab = 'MSE', col = 'red', main = "n = 3")
> lines(c(1,5,50,100), c(mse_3_1[2],mse_3_5[2], mse_3_50[2], mse_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_5_1[1],mse_5_5[1], mse_5_50[1], mse_5_100[1]), type="b",
xlab = 'theta', ylab = 'MSE', col = 'red', main = "n = 5")
> lines(c(1,5,50,100), c(mse_5_1[2],mse_5_5[2], mse_5_50[2], mse_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_10_1[1],mse_10_5[1], mse_10_50[1], mse_10_100[1]),
type="b", xlab = 'theta', ylab = 'MSE', col = 'red', main = "n = 10")
> lines(c(1,5,50,100), c(mse_10_1[2],mse_10_5[2], mse_10_50[2], mse_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_30_1[1],mse_30_5[1], mse_30_50[1], mse_30_100[1]),
type="b", xlab = 'theta', ylab = 'MSE', col = 'red', main = "n = 30")
> lines(c(1,5,50,100), c(mse_30_1[2],mse_30_5[2], mse_30_50[2], mse_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')
```

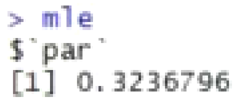
#draw graphs with fixed θ value and varying n

```
> par(mfrow=c(2,2))
> plot(c(1,2,3,5,10,30), c(mse_1_1[1],mse_2_1[1], mse_3_1[1], mse_5_1[1],
mse_10_1[1], mse_30_1[1]), type="b", ylab = 'MSE', xlab = 'n', col = 'red', main = "theta
= 1")
> lines(c(1,2,3,5,10,30), c(mse_1_1[2],mse_2_1[2], mse_3_1[2], mse_5_1[2],
mse_10_1[2], mse_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_1_5[1],mse_2_5[1], mse_3_5[1], mse_5_5[1],
mse_10_5[1], mse_30_5[1]), type="b", ylab = 'MSE', xlab = 'n', col = 'red', main = "theta
= 5")
> lines(c(1,2,3,5,10,30), c(mse_1_5[2],mse_2_5[2], mse_3_5[2], mse_5_5[2],
mse_10_5[2], mse_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_1_50[1],mse_2_50[1], mse_3_50[1], mse_5_50[1],
mse_10_50[1], mse_30_50[1]), type="b", ylab = 'MSE', xlab = 'n', col = 'red', main =
"theta = 50")
> lines(c(1,2,3,5,10,30), c(mse_1_50[2],mse_2_50[2], mse_3_50[2], mse_5_50[2],
mse_10_50[2], mse_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_1_100[1],mse_2_100[1], mse_3_100[1], mse_5_100[1],
mse_10_100[1], mse_30_100[1]), type="b", ylab = 'MSE', xlab = 'n', col = 'red', main =
"theta = 100")
> lines(c(1,2,3,5,10,30), c(mse_1_100[2],mse_2_100[2], mse_3_100[2], mse_5_100[2],
mse_10_100[2], mse_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

2c) #R code for minimizing the function

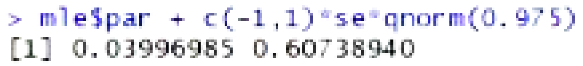
```
#create function to return negative values of the derived function

> neg.loglkhood.fn <- function(par, dat) {

        + result = length(dat) * log(par) - (par + 1) * sum(log(dat))
        + return(-result)
+}

#initialize an array with data
> x <- c(21.42, 14.65, 50.42, 28.78, 11.23)

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



```
> mle
$`par`
[1] 0.3236796
```

2d) #R code for finding the confidence interval #finding the standard error

```
> se <- (1/mle$hessian)^0.5
 > se
 [1] 0.1447525
 #finding the confidence interval
```



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