Statistical Methods for Data Science: Mini Project 3 Solved

Mini Project #: 3
Group #: 9
Names of group members: Nikita Ahuja, Bhargaw Rajnikant Patel
Contribution of each group member: Both team members worked together to solve the two problems. Both members reviewed the equations from the problems given, worked out the solutions in R, wrote the code and finished the report in a timely manner. Both partners worked equally to complete the Mini Project 3 requirements.

Question 1

Solution:

(a) Mean squared error can be determined by first setting a population parameter and then simulating the sample values, allowing the estimator value to be calculated. The estimated value of the difference squared between the estimator and parameter is now the mean squared error.

(b) The function mleandmom(n,)) mimics samples from a uniform distribution and then calculates the MLE and MOM for the supplied sample, returning a two-value array. The function msees(n,)) calls the mleandmom() function 1000 times, calculating the mean squared error using the formula E(-)2 and returning the mean squared errors for both estimators in an array format.

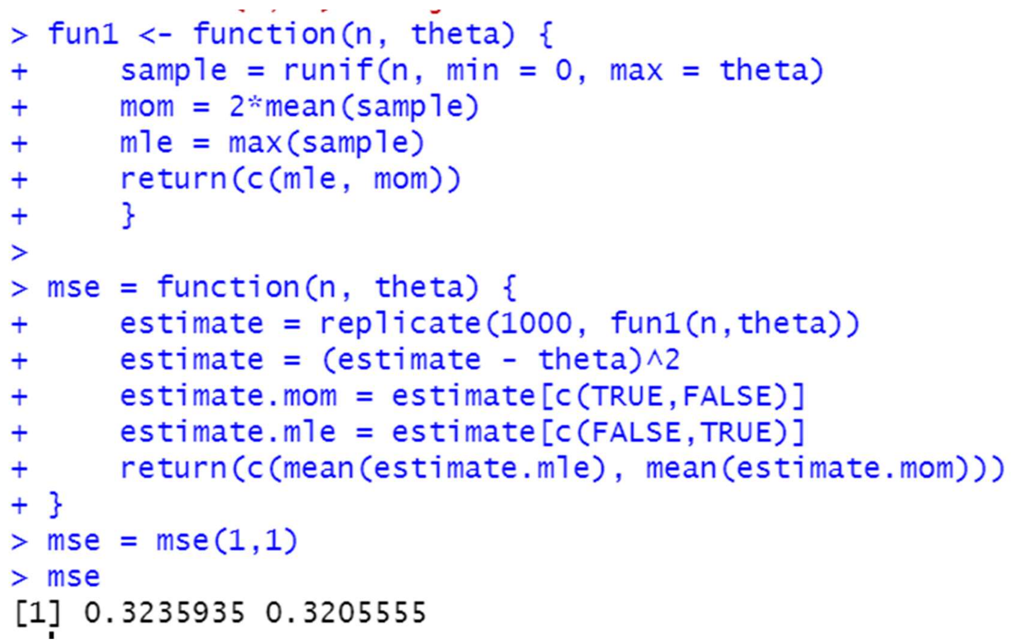
Code:

```
#Calculating the mean squared errors
#Create function to return MLE/MOM for the same sample
> fun1 <- function(n, theta) {
+ sample = runif(n, min = 0, max = theta)
+ mom = 2*mean(sample)
+ mle = max(sample)
+ return(c(mle, mom))
+}

#Create function to calculate and return mean squared errors of
MLE/MOM for 1000 replications
> mse = function(n, theta) {
+ estimate = replicate(1000, fun1(n,theta))
+ estimate = (estimates - theta)^2
+ estimate.mom = estimate[c(TRUE,FALSE)]
+ estimate.mle = estimate[c(FALSE,TRUE)]
+ return(c(mean(estimate.mle), mean(estimates.mom)))
+}
```

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

```
    > mse = mse(1,1)
```

Output:



```
> fun1 <- function(n, theta) {
+     sample = runif(n, min = 0, max = theta)
+     mom = 2*mean(sample)
+     mle = max(sample)
+     return(c(mle, mom))
+     }
>
> mse = function(n, theta) {
+     estimate = replicate(1000, fun1(n,theta))
+     estimate = (estimate - theta)^2
+     estimate.mom = estimate[c(TRUE,FALSE)]
+     estimate.mle = estimate[c(FALSE,TRUE)]
+     return(c(mean(estimate.mle), mean(estimate.mom)))
+ }
> mse = mse(1,1)
> mse
[1] 0.3235935 0.3205555
```

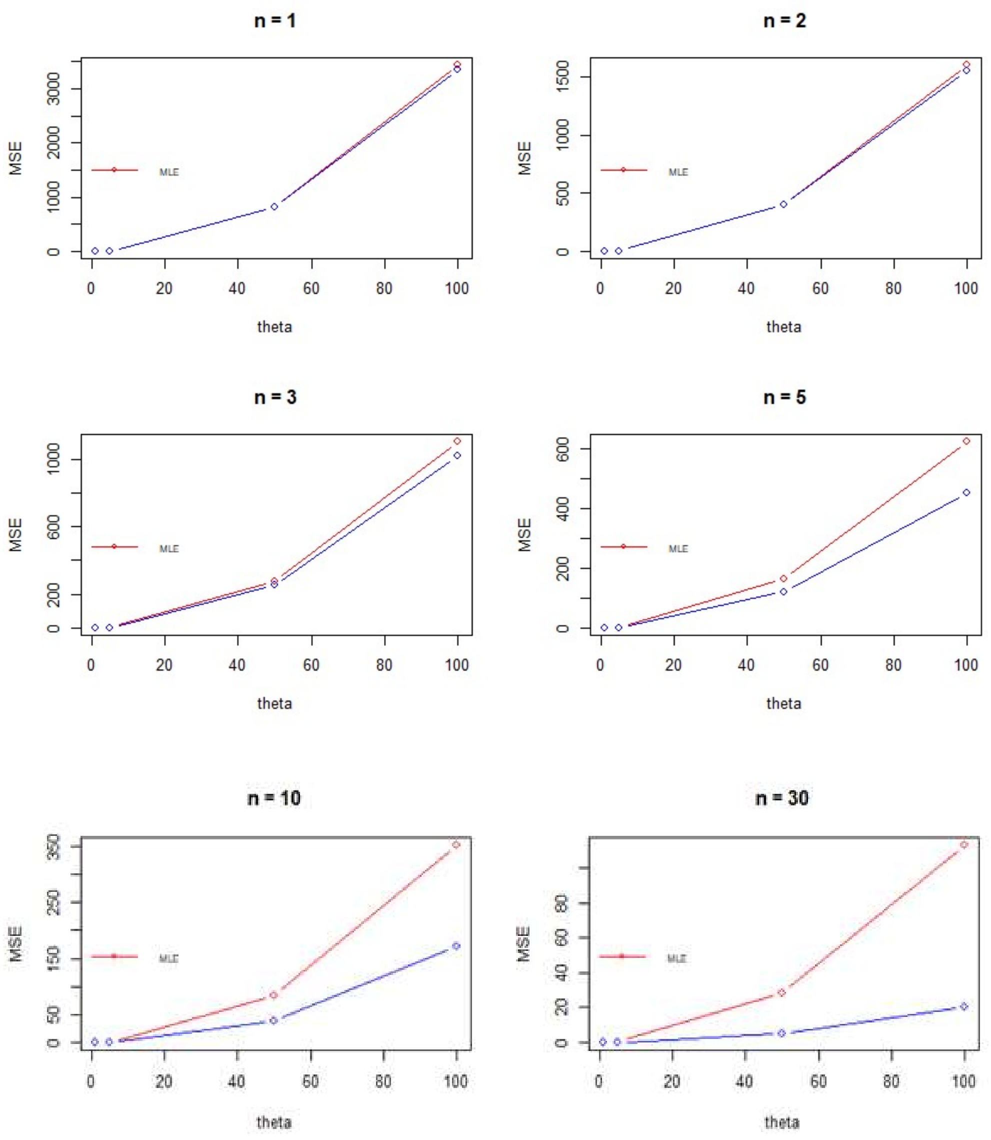
(c) The par() function is used to alter the layout of the plot output. This is done so that more than one graph can be accommodated; the syntax is par(mfrow = c(2,2)). In this example, mfrow is used to place the following graphs. As a result, the next four graphs are arranged in a 2x2 matrix.

Code:

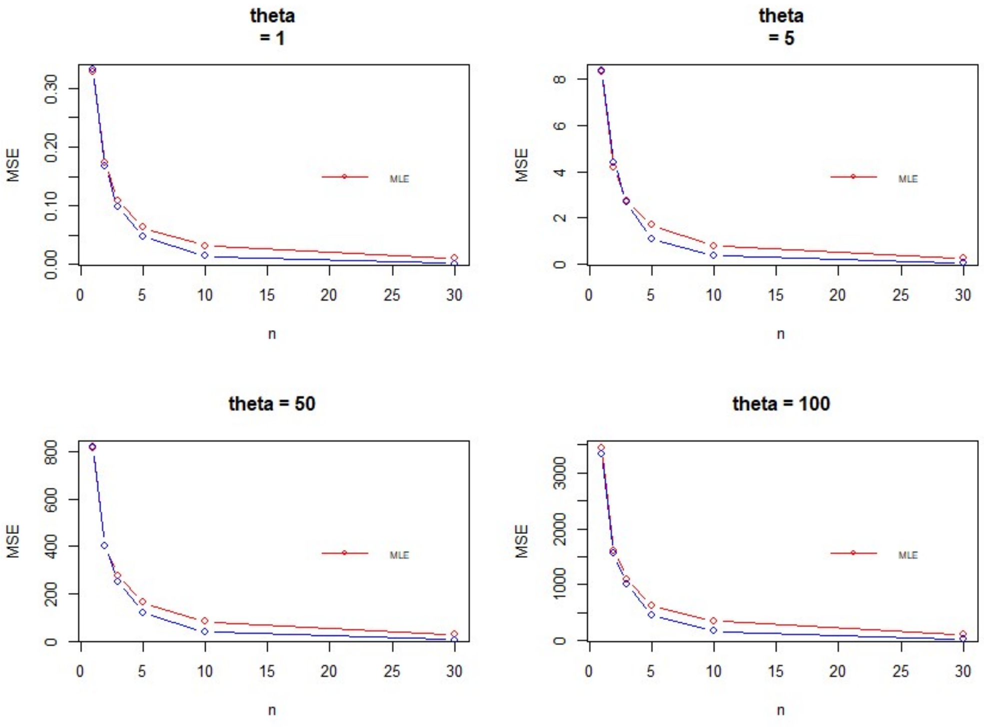
```
>mse_1_1 = mse(1,1)
> mse_1_50 = mse(1,50)
> mse_1_100 = mse(1,100)
> mse_2_1 = mse(2,1)
> mse_2_5 = mse(2,5)
> mse_2_50 = mse(2,50)
> mse_2_100 = mse(2,100)
> mse_3_1 = mse(3,1)
> mse_3_5 = mse(3,5)
> mse_3_50 = mse(3,50)
> mse_3_100 = mse(3,100)
> mse_5_1 = mse(5,1)
> mse_5_5 = mse(5,5)
> mse_5_50 = mse(5,50)
> mse_5_100 = mse(5,100)
> mse_10_1 = mse(10,1)
> mse_10_5 = mse(10,5)
> mse_10_50 = mse(10,50)
> mse_10_100 = mse(10,100)
> mse_30_1 = mse(30,1)
> mse_30_5 = mse(30,5)
> mse_30_50 = mse(30,50)
> mse_30_100 = mse(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")
>line(c(1,2,50,100), 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')
```

Output:



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) The second graph shows that regardless of what value of is fixed, the resulting graphs are highly similar. As a result, it's possible to deduce that the estimator isn't reliant on the value of. The plotted values of mean squared errors varying with fixed n are shown in Graph 1, and it is clear that we can utilize the Method of Moments estimator for small values of n (=1, 2, 3). The Maximum Likelihood Estimator is better when the n (=5, 10, 30) number grows. MLE is better for bigger n numbers, and as n grows larger, MLE becomes the preferred option. In compared to MOM, MLE is recommended since the mean squared error is lower for the same value of n.

Problem 2:

Solution:

Probability density function: $f(x) = \frac{\theta}{x^{\theta+1}},\ x \geq 1$

And $f(x) = 0, x < 1$

Unknown parameter = θ

X_1, ….., X_n is the given SRS (Simple random sample of size n from the population)

a) We have to derive an expression for the Maximum Likelihood Estimator (MLE) of θ.

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

   Take the log of both sides, $\log L(\theta) = \log \prod_{i=1}^{n} \frac{\theta}{{x_i}^{\theta+1}}$

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

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

   Now we take the partial derivative of the above equation and get:

   = n / θ - $\sum_{i=1}^{n} \log(x_i)$

   We equate this result to zero to find the MLE.

   n / θ - $\sum_{i=1}^{n} \log(x_i) = 0$

   n / θ = $\sum_{i=1}^{n} \log(x_i)$

   $\hat{\theta}_{MLE}$ = n / $\sum_{i=1}^{n} \log(x_i)$

b) Given: n = 5, x1 = 21.72, x2 = 14.65, x3 = 50.42, x4 = 28.78, x5 = 11.23

   Substituting the values in the MLE equation,

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

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

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

c) We have to maximize the log-likelihood function using the optim() function in R. Since we have to maximize the negative of the function, we negate it and instead minimize the function in R.

   optim(par, fn, ….) is used where par refers to the values of the parameters that we want to optimize, fn is the function being minimized.

   Firstly, we define the log of the likelihood function in R and return the negative result.

```
> neg.loglikelihood.fn <- function(par, dat) {
+     result = length(dat) * log(par) - (par + 1) * sum(log(dat))
+     return(-result)
+ }
```

We create an array of the sample values and then execute the optim() function in R to minimize the derived function's negative value.

```
> x <- c(21.42, 14.65, 50.42, 28.78, 11.23)
> mle <- optim(par=0.926, fn=neg.loglikelihood.fn, method="L-BFGS-B",
  hessian=TRUE,lower=0.01, dat=x)
> mle
$par
[1] 0.3236796
```

Therefore, it is proved through the R code that the estimate is 0.3236.

d) To find an approximate standard error of the maximum likelihood estimate and an approximate 95 % confidence interval for θ.

The formula for standard error is: SE($\hat{\theta}$) approximates to $\sqrt{\hat{I}^{-1}}$, where $\hat{I}$ is the Hessian function.

From the R code, we can conclude that the standard error is 0.1447525.

We know that 1 – α = 0.95, α = 0.05, α / 2 = 0.025,

So, 1 – α / 2 = 0.975

The confidence interval formula is : $\hat{\theta}$ = Z α/2 x SE($\hat{\theta}$)

We find the Z α/2 value by using the qnorm() function.

From the R code, our confidence interval is: (0.03996985, 0.6073890).

```
> se <- (1/mle$hessian)^0.5
> se
          [,1]
[1,] 0.1447525
>
> # To find the confidence interval
> mle$par + c(-1,1) * se * qnorm(0.975)
[1] 0.03996985 0.60738940
```

Therefore, we conclude that the true estimated value lies within the confidence interval 95 % of the times out of the 100 times we try to get the population estimation.