STATISTICAL METHODS FOR DATA SCIENCE Mini-Project 3

Duo Group #23

Members: Hima Sri Tipirineni

Nithin Pingili

Contribution of each team member:

Hima Sri and Nithin worked together to complete both the questions. Collaborated to learn R and then worked on calculating the mean squared errors of the maximum likelihood estimator and method of moments estimator. Both worked together to answer the questions and report all the findings. Hima Sri wrote R code and annotated the code and Nithin worked to check the accuracy of the R code and added the observations . Both worked efficiently to complete all sections of the project.

Question 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 theta is defined. For the comparison, we will focus on n = 1, 2, 3, 5, 10, 30 and theta = 1, 5, 50, 100.

(a) Explain how you will compute the mean squared error of an estimator using Monte Carlo simulation.

Solution :

Mean squared error can be calculated by first setting a population parameter then simulating the sample values . Repeat the process of simulating a sample and computing mean from a simulating sample a large number of times and compute the average of the squared deviations between theta and theta (hat).

R code :

```
### Function to calculate mean squared error (MSE) for MLE and MOM ###
Estimatorfunction = function(n, theta) {
  sample = runif(n,min = 0, max = theta);
## Montecarlo simulation for generating a uniform distribution ##
  mme = 2 * mean(sample);
## Calculating mean ##
  mle = max(sample);
  return (c(mme,mle));
```

```
}

MSEfunction = function(n , theta) {
 sampleestimators = replicate(nsim,Estimatorfunction(n,theta));
 error1 = (sampleestimators - theta)^2;
 return(rowMeans(error1));
}
```

```
>
>
>
> ######## n = 1,2,3,5,10,30 ; theta = 1,5,50,500 #########
>
> ### Function to calculate mean squared error (MSE) for MLE and MOM ###
>
> Estimatorfunction = function(n, theta) {
+   sample = runif(n,min = 0, max = theta);
+   mme = 2 * mean(sample);
+   mle = max(sample);
+   return (c(mme,mle));
+ }
>
> MSEfunction = function(n , theta) {
+   sampleestimators = replicate(nsim,Estimatorfunction(n,theta));
+   error1 = (sampleestimators - theta)^2;
+   return(rowMeans(error1));
+ }
>
```

(b) For a given combination of (n,theta), compute the mean squared errors of both theta_1 and theta_2 using Monte Carlo simulation with N = 1000 replications. Be sure to compute both estimates from the same data.

Solution :

```
## For 1000 simulations and for n = 10, theta = 50 ##

MSEfunction = function(n , theta) {
 sampleestimators = replicate(1000,Estimatorfunction(n,theta));
 error1 = (sampleestimators - theta)^2;
 return(rowMeans(error1));
}
theta_hat = MSEfunction(10, 50);
theta_hat
```

```
>
> ## For 1000 simulations and for n = 10, theta = 50 ##
>
> MSEfunction = function(n , theta) {
+   sampleestimators = replicate(1000,Estimatorfunction(n,theta));
+   error1 = (sampleestimators - theta)^2;
+   return(rowMeans(error1));
+ }
> theta_hat = MSEfunction(10, 50);
> theta_hat
[1] 80.89973 36.63707
>
```

(c) Repeat (b) for the remaining combinations of (n, theta). Summarize your results graphically.

Solution :

R code :

```
## For various combinations of n and theta calculating mse ##
n = c(1,2,3,5,10,30)
theta = c(1,5,50,100)
nsim = 1000

n_len = length(n)
theta_len = length(theta)
#set.seed(1000)
mse_mme_theta1 = matrix(NA,nrow = n_len, ncol = theta_len)
mse_mle_theta2 = matrix(NA, nrow = n_len, ncol = theta_len)

for(i in 1:n_len) {
 for(j in 1:theta_len) {
  val = MSEfunction(n[i],theta[j])
  mse_mme_theta1[i,j] = val[1]
  mse_mle_theta2[i,j] = val[2]
 }
}

mse_mme_theta1

mse_mle_theta2

## Plotting the results graphically ###

par(mfrow = c(2,2))
for(i in 1:theta_len){
 plot(n,mse_mme_theta1[,i],type="l",lty=1,ylim =
c(0,max(mse_mme_theta1[,i],mse_mle_theta2[,i])),main=paste0("theta=",theta[i]),
     ylab = "MSE",col='red')
 lines(n,mse_mle_theta2[,i],lty = 2,col = 'blue')
```

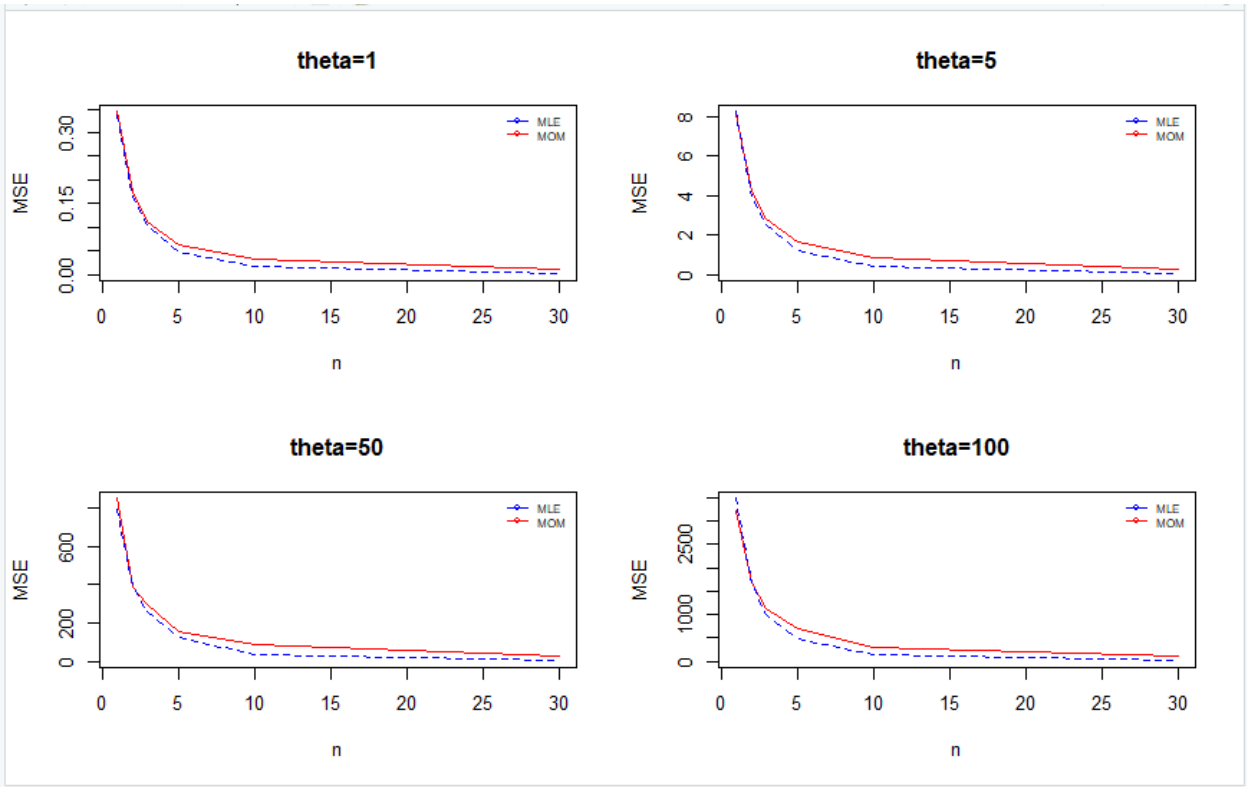
```
 legend("topright",legend = c("MLE","MOM"),col = c('blue','red'),text.col = c('black','black'),lty = 1, pch =
1, inset = 0.01,ncol = 1, cex = 0.6,bty = 'n')
}


## Plotting the results graphically with different values of n ###

par(mfrow = c(3,3))
for(i in 1:n_len){
 plot(theta,mse_mme_theta1[i,],type="l",lty=1,ylim =
c(0,max(mse_mme_theta1[i,],mse_mle_theta2[i,])),main=paste0("n=",n[i]),
     ylab = "MSE",col='red')
 lines(theta,mse_mle_theta2[i,],lty = 2,col = 'blue')
legend("topleft",legend = c("MLE","MOM"),col = c('blue','red'),text.col = c('black','black'),lty = 1, pch = 1,
       inset = 0.01,ncol = 1, cex = 0.6,bty = 'n')
}
```

```
> ## For various combinations of n and theta calculating mse ##
> n = c(1,2,3,5,10,30)
> theta = c(1,5,50,100)
> nsim = 1000
>
> n_len = length(n)
> theta_len = length(theta)
> #set.seed(1000)
> mse_mme_theta1 = matrix(NA,nrow = n_len, ncol = theta_len)
> mse_mle_theta2 = matrix(NA, nrow = n_len, ncol = theta_len)
>
> for(i in 1:n_len) {
+   for(j in 1:theta_len) {
+     val = MSEfunction(n[i],theta[j])
+     mse_mme_theta1[i,j] = val[1]
+     mse_mle_theta2[i,j] = val[2]
+   }
+ }
>
> mse_mme_theta1
           [,1]      [,2]      [,3]      [,4]
[1,] 0.34447577 8.0945523 851.61875 3235.8794
[2,] 0.17685835 4.1959170 389.23721 1678.6222
[3,] 0.11229112 2.8221901 295.05415 1120.4983
[4,] 0.06367996 1.6164819 154.05493  692.6549
[5,] 0.03350799 0.8766067  87.73876  310.4746
[6,] 0.01077579 0.2560750  28.19444  102.6632
>
> mse_mle_theta2
            [,1]      [,2]       [,3]       [,4]
[1,] 0.336045389 8.2294167 796.052978 3486.22003
[2,] 0.164281402 3.9861006 397.233577 1676.37759
[3,] 0.104518152 2.5051727 260.681495 1009.49802
[4,] 0.046646430 1.1905394 122.595923  478.50941
[5,] 0.015790569 0.4054639  34.797260  136.49866
[6,] 0.002041313 0.0424597   5.105977   18.70251
>
```

```
>
> ## Plotting the results graphically ###
>
> par(mfrow = c(2,2))
> for(i in 1:theta_len){
+   plot(n,mse_mme_theta1[,i],type="l",lty=1,ylim = c(0,max(mse_mme_theta1[,i],mse_mle_theta2[,i])),main=paste0("theta=",thet
a[i]),
+        ylab = "MSE",col='red')
+   lines(n,mse_mle_theta2[,i],lty = 2,col = 'blue')
+   legend("topright",legend = c("MLE","MOM"),col = c('blue','red'),text.col = c('black','black'),lty = 1, pch = 1,
+          inset = 0.01,ncol = 1, cex = 0.6,bty = 'n')
+ }
>
```

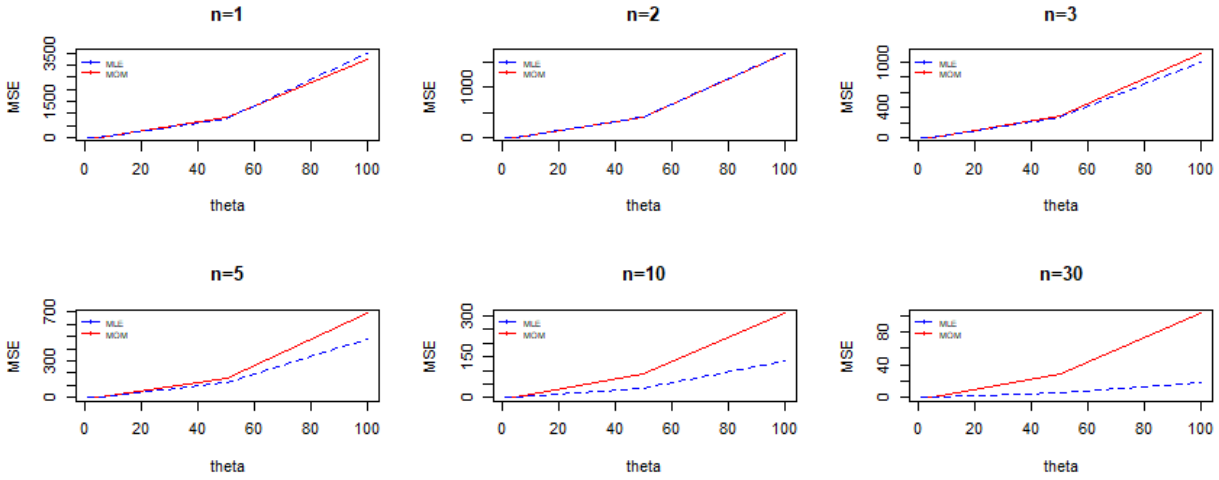
```
>
> ## Plotting the results graphically with different values of n ###
>
> par(mfrow = c(3,3))
> for(i in 1:n_len){
+   plot(theta,mse_mme_theta1[i,],type="l",lty=1,ylim = c(0,max(mse_mme_theta1[i,],mse_mle_theta2[i,])),main=paste0("n=",n
[i]),
+        ylab = "MSE",col='red')
+   lines(theta,mse_mle_theta2[i,],lty = 2,col = 'blue')
+   legend("topleft",legend = c("MLE","MOM"),col = c('blue','red'),text.col = c('black','black'),lty = 1, pch = 1,
+          inset = 0.01,ncol = 1, cex = 0.6,bty = 'n')
+ }
> |
```

Graph 1 : Mean Squared errors of MLE and MOM , n with fixed ɵ.



Here , red line indicates Method of Moments estimator, blue line indicates Maximum likelihood estimator.

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



Here , red line represents Method of Moments estimator, blue line represents Maximum likelihood estimator

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

Solution :

The MSE of both estimators decreases as n increases. This is because the estimators become more accurate as n increases. For different values of theta, the graphs are very similar because the MSE curves are identical. So, the estimator does not depend on theta. When n is 1 and 2, the method of moments estimator is better because the MME line is at or below the MLE line when n is 1 and 2. When n is 3, 5, 10 and 30, the maximum likelihood estimator is better because the MLE line is below the MME line when n is 3, 5, 10 and 30. For small values of n(n = 1 or 2), method of moments estimator is better. However, for large values of n(n > 5) the maximum likelihood estimator is better. The better estimator depends on n because the MME and MLE curves are different each n value.

2) Suppose the lifetime, in years, of an electronic component can be modeled by a continuous random variable with probability density function

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

where $\theta > 0$ is an unknown parameter. Let $X_1, \ldots, X_n$ be a random sample of size n from this population.

a) Derive an expression for maximum likelihood estimator of theta.

Solution :

The maximum likelihood estimator of theta is calculated as below :

Given

The Likelihood function is :

$L(\theta) = \pi_{i=1}^{n} (\theta / x_i^{\theta+1})$

The log likelihood function is :

$\log(L(\theta)) = \log (\pi_{i=1}^{n} (\theta / x_i^{\theta+1})$

$= \log(\theta^n * \pi_{i=1}^{n} (1 / x_i^{\theta+1}))$

$= \log(\theta^n) + \Sigma_{i=1}^{n} \log x_i^{-(\theta+1)}$

$= n \log \theta - (\theta + 1) \Sigma_{i=1}^{n} \log x_i$

$\log(L(\theta)) = n \log \theta - \theta \Sigma_{i=1}^{n} \log x_i - \Sigma_{i=1}^{n} \log x_i$

Differentiating on both sides wrt to θ;

$d/d\theta [\log(L(\theta))] = n/\theta - \Sigma_{i=1}^{n} \log x_i - 0$ [Since the third term is independent of θ].

Equating the differentiated value to 0 to get the maximum loglikelihood :

$n/\theta - \Sigma_{i=1}^{n} \log x_i = 0$

$n/\theta = \Sigma_{i=1}^{n} \log x_i$

$\theta = n / \Sigma_{i=1}^{n} \log x_i$

Hence, the maximum likelihood estimator of θ is **$n / \Sigma_{i=1}^{n} \log x_i$**

b) Suppose n = 5 and the sample values are x1 = 21.72; x2 = 14.65; x3 = 50.42; x4 =28.78; x5 = 11.23. Use the expression in (a) to provide the maximum likelihood estimate for θ based on these data.

Solution :

Given that

$n = 5;$

$x_1 = 21.72;$

$x_2 = 14.65;$

$x_3 = 50.42;$

$x_4 = 28.78;$

$x_5 = 11.23;$

$\theta = n / \Sigma_{i=1}^{n} \log x_i$

$\theta = 5 /(\log(21.72) + \log(14.65) + \log(50.42) + \log(28.78) + \log(11.23))$

$\theta = 5 / (3.078 + 2.684 + 3.920 + 3.359 + 2.418)$

$\theta = 5 / 15.459$

$\theta = 0.323$

c) Even though we know the maximum likelihood estimate from (b), use the data in (b) to obtain the estimate by numerically maximizing the log-likelihood function using optim function in R. Do your answers match?

Solution :

R code :

```
## defining the function obtained in question 2(a) ##
negative_log_function = function(theta, value){
  result = length(value) * log(theta) - (theta + 1) * sum(log(value));
  return (-result);
}

## executing the optim function to minimize the negative log function ##

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

mle_estimator = optim(par = 0.3, fn = negative_log_function, method = "L-BFGS-B",lower = (10
^ (-10)), hessian = TRUE,value =given_data)

mle_estimator$par
```

```
>
> ## defining the function obtained in question 2(a) ##
> negative_log_function = function(theta, value){
+   result = length(value) * log(theta) - (theta + 1) * sum(log(value));
+   return (-result);
+ }
>
> ## executing the optim function to minimize the negative log function ##
>
> given_data = c(21.72,14.65,50.42,28.78,11.23)
>
> mle_estimator = optim(par = 0.3, fn = negative_log_function, method = "L-BFGS-B",lower = (10 ^ (-10)), hessian = TRUE,value
 =given_data)
>
> mle_estimator$par
[1] 0.3233884
>
```

Yes, the answers match.

d) Use the output of numerical maximization in (c) to provide an approximate standard error of the maximum likelihood estimate and an approximate 95% confidence interval for theta. Are these approximations going to be good? Justify your answer.

Solution :
R code :
To get the approximate standard error and 95% confidence interval for theta

standar_error = sqrt(1/mle_estimator$hessian)
standar_error

confidence_interval = mle_estimator$par + c(-1,1)*standar_error*qnorm(0.975)
confidence_interval

```
>
> #### To get the approximate standard error and 95% confidence interval for theta #####
>
> standar_error = sqrt( 1/mle_estimator$hessian )
> standar_error
          [,1]
[1,] 0.1446223
>
> confidence_interval = mle_estimator$par + c(-1,1)*standar_error*qnorm(0.975)
> confidence_interval
[1] 0.03993389 0.60684301
>
```

The approximations are based on sample size(n = 5). By large sample properties of MLE, the estimator follows asymptotically normal distribution if n is large. But in this case the n value is very small . Hence the approximations cannot be considered good completely.
The 95% confidence interval for theta is a good approximation because the actual estimate of theta is between 0.0399 and 0.6068 95% of the time.