**MINI PROJECT #2**

**Name:** Omkar Nandkumar Dixit (ond170030)

**Name of group members:** Sushrut Patnaik (sxp175331)

**Contributions of each group members:**

We discussed both the questions together, but Omkar coded the first question and Sushrut coded the second question.

**Section 1:**

**Question 1 –**

1. Mean squared error can be found out using Monte Carlo simulation. In the first Step we need to draw random samples from a Uniform distribution. Then we need to calculate the MLE which is equal to the maximum of the sample and the MOME which is twice the mean of the sample. The next step would be to calculate the square of the difference of the actual value and the estimator. The process is repeated several times and squared error is computed each time. Mean of the squared errors is noted.
2. For n=1, θ=1, we get the following values for MSE.

MSE for MLE is 0.333392334, MSE for MOME is 0.33233603, We can see the values from MSE matrices for MLE and MOME

The rows in the matrices represent n = 1, 2, 3, 5, 10, 30

The columns represent theta = 1,5,50,100

> mseofmle

[,1] [,2] [,3] [,4]

[1,] 0.333392334 8.68811207 862.79636 3421.6690

[2,] 0.157267277 4.13823790 417.34638 1615.1136

[3,] 0.102622070 2.65179596 242.56684 977.0988

[4,] 0.049821948 1.18015794 117.99735 443.1750

[5,] 0.014815606 0.34764120 37.28650 151.4734

[6,] 0.002134135 0.04709599 5.29231 20.5792

> mseofmome

[,1] [,2] [,3] [,4]

[1,] 0.33233603 8.7820652 848.15702 3628.8080

[2,] 0.16285111 4.2199047 409.83348 1629.5106

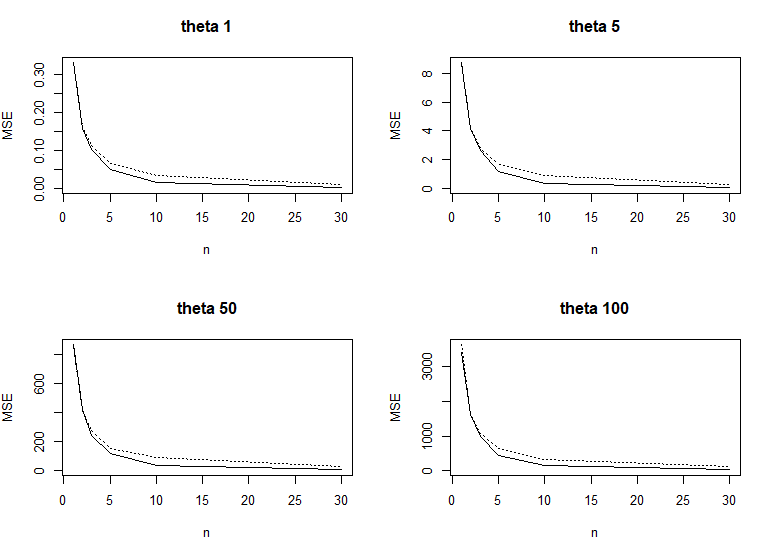
[3,] 0.11210381 2.7771968 273.42616 1075.1787

[4,] 0.06447459 1.6607492 155.35389 638.1882

[5,] 0.03398797 0.8692844 87.32239 326.1817

[6,] 0.01103115 0.2675745 26.72338 110.8447

1. The MSE for all combinations of (n, θ) are given the 2 matrices in (b). The results are summarized graphically. The solid curves represent the MSE of the MLE of θ while the dotted curves represent the MSE of MOME of θ.



1. We see that MSE decreases as the value of n increases. This is because as n increases accuracy will also increase. We can also see that as theta increases the MSE also increases. For smaller values of n, the MSEs of both the estimators are equal but as the value of n increases MLE has slightly less MSE than the MSE of MOME. For values of n<5 MSE of both the estimators are almost equal but for n>=5 MLE has less MSE than MOME. Therefore, for any value of n and theta, MLE is better than MOME.

**Question 2 –**

1. We have been given a pdf: f(x)=θ/x(θ+1)

So, the likelihood function is L(θ) = θ/xi(θ+1)

To find the parameter estimate we need to apply log on both sides and differentiate with respect to θ and equate it to 0.

Log(L(θ) = Log( θ/xi(θ+1))

= log( θn \*xi-( θ+1) )

= nlog(θ) – (θ+1)

∂ log L(θ)/ ∂θ=0

∂ / ∂θ [= 0

n/ θ - = 0

θ = n/[

1. Given n = 5 and the sample values are x1 = 21.72; x2 = 14.65; x3 = 50.42; x4 = 28.78; x5 =11.23. Substituting these values in the expression derived for MLE in question (a)

gives us the following.

θ = 5/(log(21.72) + log(14.65) + log(50.42) + log(28.78) + log(11.23))

θ = 0.3233874

1. Now we find the MLE using the optim function in R with the given data in (b). We get the following value for θ = 0. 323387. The value matches the value we calculated in the question (b).

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

#define the function given by the pdf in question

mletheta<- function(x,a)

{

return(a/x^(a+1))

}

#We take negative of log-likelihood function because by default, Optimization routines do a minimization.

# Negative of log-likelihood function

neg.loglik.fun<-function(dat,par){

result<-sum(log(mletheta(dat,par)))

return(-result)}

# Minimize -log (L), i.e., maximize log (L)

ml.est<-optim(par=1, fn=neg.loglik.fun, method ="BFGS",hessian=TRUE, dat=input)

#find the parameter estimate

> ml.est$par

[1] 0.323387

1. The Standard error of the estimate was found to be 0.1446217 and the Confidence interval is as follows: Upper CI 0.6068456 and lower CI 0.03992851. The approximations cannot be considered as good because the sample size is very small.

#Standard error of the estimate

> sqrt(diag(solve(ml.est$hessian)))

[1] 0.1446217

> #95% confidence Interval of the estimate

> upper<-ml.est$par + 1.96 \* sqrt(diag(solve(ml.est$hessian)))

> lower<-ml.est$par - 1.96 \* sqrt(diag(solve(ml.est$hessian)))

>

> interval<-data.frame(upper=upper, lower=lower)

> interval

upper lower

1 0.6068456 0.03992851

**Section 2: R Code**

**Question 1 –**

# function to find MSE

mse<-function(n, theta)

{

#function to find both MLE and MOME

computestimate<-function(n, theta)

{

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

mle<-max(x)

mome<-2\*mean(x)

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

}

# replicate 1000 times

estimate<-replicate(1000, computestimate(n, theta))

# return MSE value(MLE and MOME) for a particular n and theta

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

}

#########################################################

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

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

set.seed(1)

#get length of n and theta matrix

nlength=length(n)

thetalength=length(theta)

#create two (n x theta) matrices to store MSE of MLE and MOME for every combination of n and theta

mseofmle<-matrix(nrow = nlength, ncol = thetalength)

mseofmome<-matrix(nrow = nlength, ncol = thetalength)

for(i in 1:nlength)

{

for(j in 1:thetalength)

{

#call mse function for n=n[i] and theta=theta[j]

result<-mse(n[i],theta[j])

mseofmle[i,j]<-result["mle"]

mseofmome[i,j]<-result["mome"]

}

}

#dislay the MSE matrices for MLE and MOME for all Combinations of n, theta

mseofmle

mseofmome

#plotting the MSE graphs(MLE and MOME)on the same plot for each value of theta

par(mfrow=c(2, 2))

for(i in 1:thetalength) {

#MLE plot in solid line

plot(n, mseofmle[,i], lty="solid",type="l", main=paste("theta",theta[i]), ylab="MSE", ylim=c(0, max(mseofmome[, i], mseofmle[, i])))

#MOME plot in dotted lines

lines(n, mseofmome[,i],lty="dotted")

}

**Question 2 –**

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

#define the function given by the pdf in question

mletheta<- function(x,a)

{

return(a/x^(a+1))

}

# Negative of log-likelihood function

neg.loglik.fun<-function(dat,par){

result<-sum(log(mletheta(dat,par)))

return(-result)}

# Minimize -log (L), i.e., maximize log (L)

ml.est<-optim(par=1, fn=neg.loglik.fun, method ="BFGS",hessian=TRUE, dat=input)

> #find the parameter estimate

> ml.est$par

[1] 0.323387

#Standard error of the estimate

> sqrt(diag(solve(ml.est$hessian)))

[1] 0.1446217

> #95% confidence Interval of the estimate

> upper<-ml.est$par + 1.96 \* sqrt(diag(solve(ml.est$hessian)))

> lower<-ml.est$par - 1.96 \* sqrt(diag(solve(ml.est$hessian)))

>

> interval<-data.frame(upper=upper, lower=lower)

> interval

upper lower

1 0.6068456 0.03992851