Mini project #1

Group Member: Chaoran Li, Wenting Wang

Contribution of each member:

Firstly, we discussed the mathematical models and code details together. Then, we divided the project into two part and finished our respective work. Wenting Wang mainly worked on Q2 while Chaoran Li worked on Q1. Then, we merged our code and solution into one report. Each member makes contribution to each sub task of this project and combines all to finish this project, as the details shown in table 1.

	Question1	Question2
Chaoran li	30%	70%
Wenting wang	70%	30%

Table 1: Member contribution table

Question1:

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

Solution:

1) (Let's start from one single trial.) we will firstly generate a population X~Uniform(0, θ) with a sample size of n;
2) We will compute the $\hat{\theta}$ based on the estimator and the population $X_1, X_2, \cdots, X_n$;

3) Since θ is known, we can get $(\hat{\theta} - \theta)^2$ easily;

4) Repeat 1) to 3) multiple times;

5) Then, we have $MSE = E[(\hat{\theta} - \theta)^2] = mean\left((\hat{\theta} - \theta)^2\right)$.

(b) For a given combination of (n, θ), compute the mean squared errors of both θ1 and θ2 using Monte Carlo simulation with N = 1000 replications. Be sure to compute both estimates from the same data.

Solution:

R code:

```r
# (b) For a given combination of (n, θ), compute the mean squared errors of both
# θ1 and θ2 using Monte Carlo simulation with N = 1000 replications. Be sure to
# compute both estimates from the same data.

n = 30
theta = 1
N = 1000
estimator <- function(n, theta){
  xs = runif(n, min=0, max=theta)
  c((max(xs)-theta)^2, (2*mean(xs)-theta)^2)
}
MSE <- function(N, n, theta){
  res <- replicate(N, estimator(n, theta))
  c(mean(res[1,]), mean(res[2,]))
}
MSE(N, n, theta)# (MLE, MME)

```

Result:

```
> MSE(N, n, theta)# (MLE, MME)
[1] 0.00220509 0.01101800
>
```

Hence, for given N = 1000, n = 30 and θ = 1, I get MSE_{MLE} = 0.00220509 and MSE_{MME} = 0.01101800.

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

Solution:

R code:

```r
# (c) Repeat (b) for the remaining combinations of (n,θ). Summarize your results
# graphically.

ns = c(1, 2, 3, 5, 10, 30)
thetas = c(1, 5, 50, 100)
res.mle = matrix(0, ncol=length(thetas), nrow=length(ns))
res.mme = matrix(0, ncol=length(thetas), nrow=length(ns))
for (i in seq_along(ns)){
  for (j in seq_along(thetas)){
    tmp = MSE(N, ns[i], thetas[j])
    res.mle[i,j] = tmp[1]
    res.mme[i,j] = tmp[2]
  }
}

```

Firstly, we will get two matrixes. One for MLE and one for MME. Then, we will control factors here. We would change only one factor in each diagram.

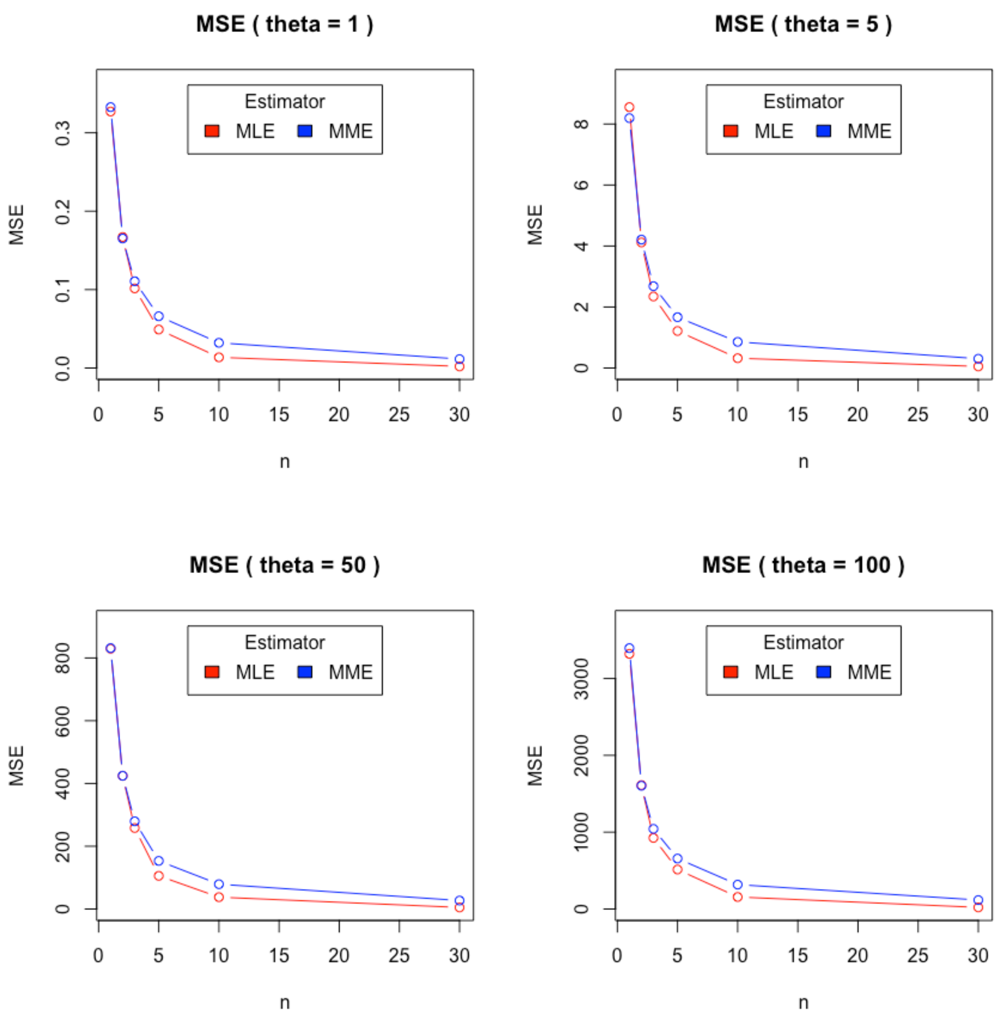
If we keep the value of θ and change n.

R code:

```r
# different n, one theta
for (i in seq_along(ns)){
  lim = c(0, max(max(res.mle[i,]), max(res.mme[i,]))*1.1)
  plot(x=thetas, y=res.mle[i,], type="b", xlab="theta", ylab="MSE", ylim=lim,
       col="red", main=paste("MSE ( n =",ns[i],")"))
  lines(x=thetas, y=res.mme[i,], type="b", col="blue")
  legend("top", inset=.05, title="Estimator", c("MLE","MME"),
       fill=c("red", "blue"), horiz=TRUE)
}
# different theta, one n
for (j in seq_along(thetas)){
  lim = c(0, max(max(res.mle[,j]), max(res.mme[,j]))*1.1)
  plot(x=ns, y=res.mle[,j], type="b", xlab="n", ylab="MSE", ylim=lim,
       col="red", main=paste("MSE ( theta =",thetas[j],")"))
  lines(x=ns, y=res.mme[,j], type="b", col="blue")
  legend("top", inset=.05, title="Estimator", c("MLE","MME"),
       fill=c("red", "blue"), horiz=TRUE)
}

```

Diagrams:



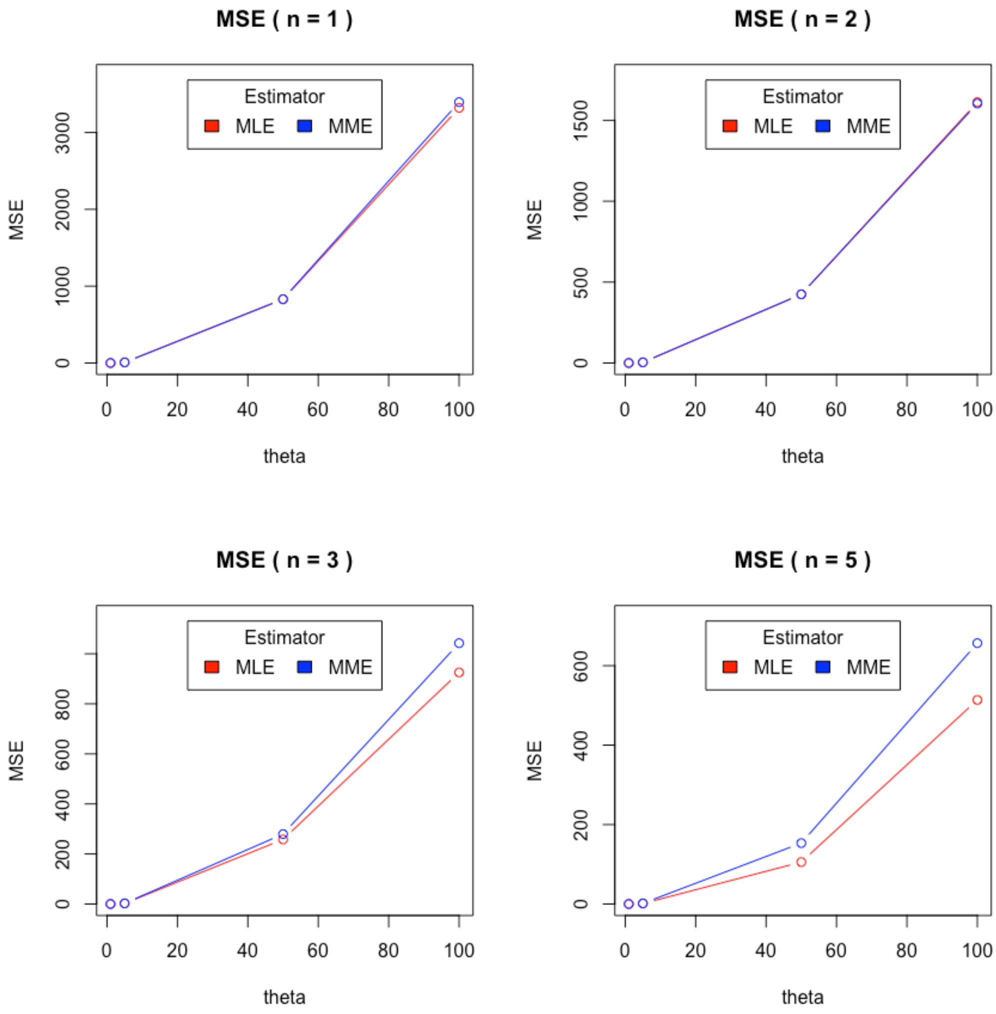
If we keep the value of n and change θ.

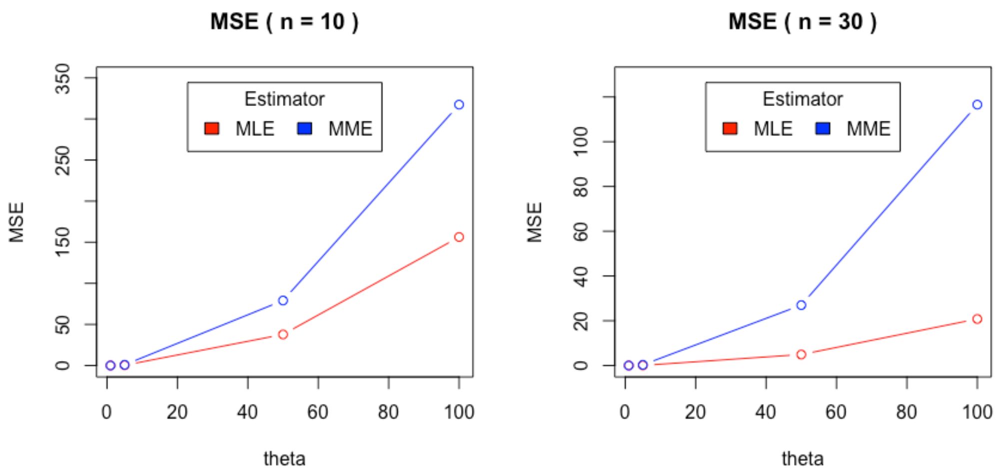
R code:

```r
# different n, one theta
for (i in seq_along(ns)){
  lim = c(0, max(max(res.mle[i,]), max(res.mme[i,]))*1.1)
  plot(x=thetas, y=res.mle[i,], type="b", xlab="theta", ylab="MSE", ylim=lim,
       col="red", main=paste("MSE ( n =",ns[i],")"))
  lines(x=thetas, y=res.mme[i,], type="b", col="blue")
  legend("top", inset=.05, title="Estimator", c("MLE","MME"),
       fill=c("red", "blue"), horiz=TRUE)
}
# different theta, one n
for (j in seq_along(thetas)){
  lim = c(0, max(max(res.mle[,j]), max(res.mme[,j]))*1.1)
  plot(x=ns, y=res.mle[,j], type="b", xlab="n", ylab="MSE", ylim=lim,
       col="red", main=paste("MSE ( theta =",thetas[j],")"))
  lines(x=ns, y=res.mme[,j], type="b", col="blue")
  legend("top", inset=.05, title="Estimator", c("MLE","MME"),
       fill=c("red", "blue"), horiz=TRUE)
}

```

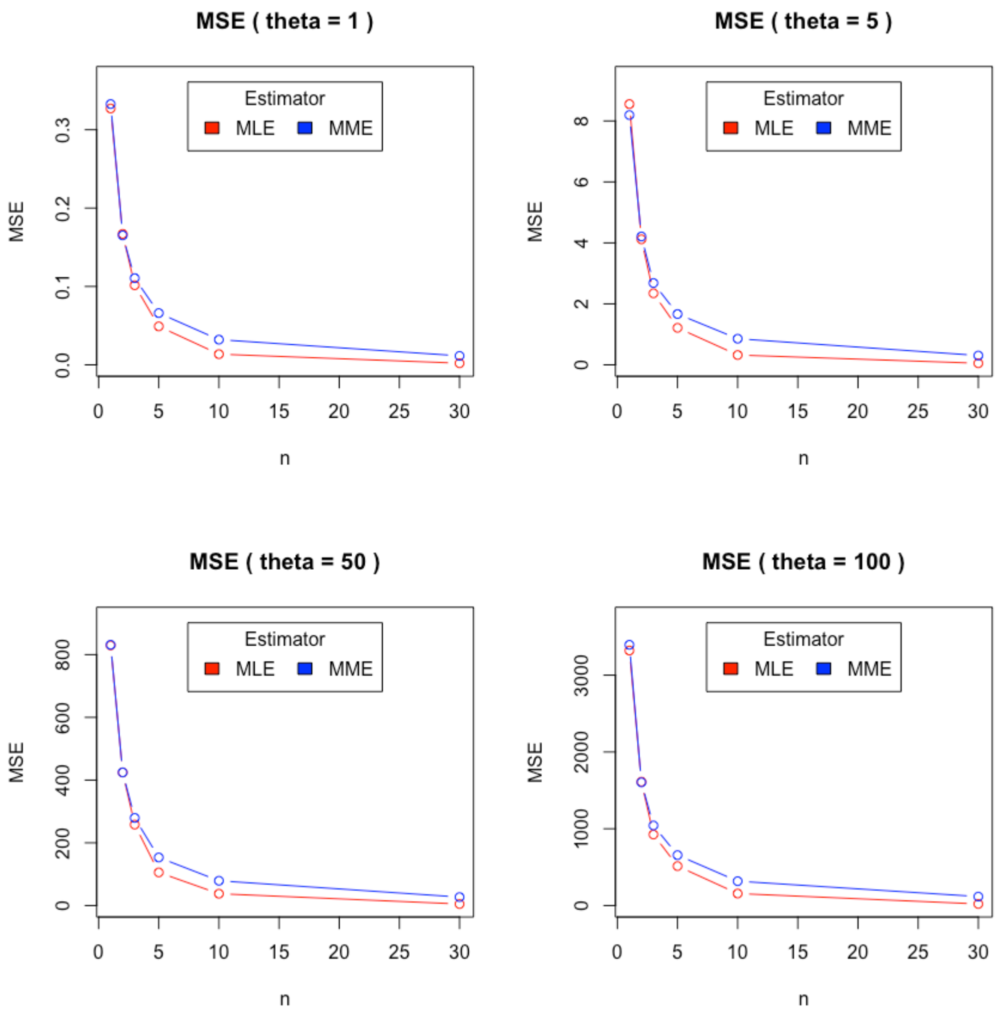
Diagrams:



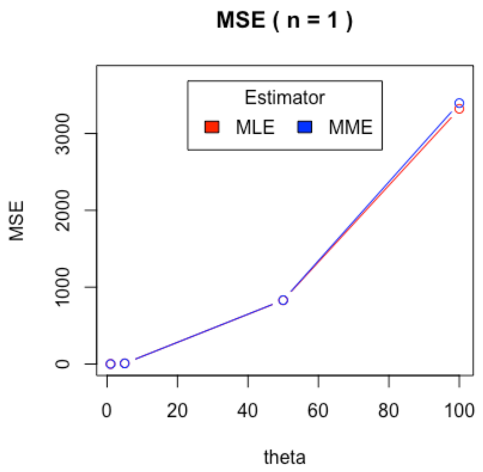


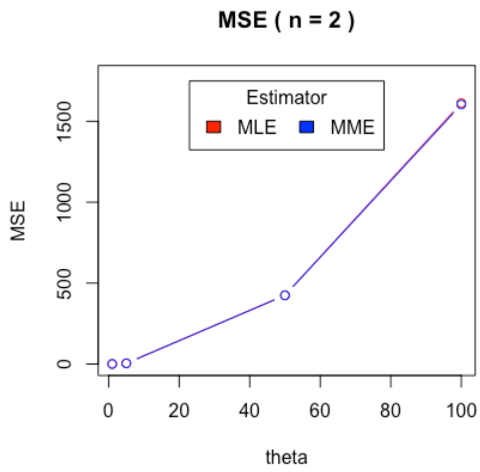
(d) Based on (c), which estimator is better? Does the answer depend on n or θ? Explain. Provide justification for all your conclusions.

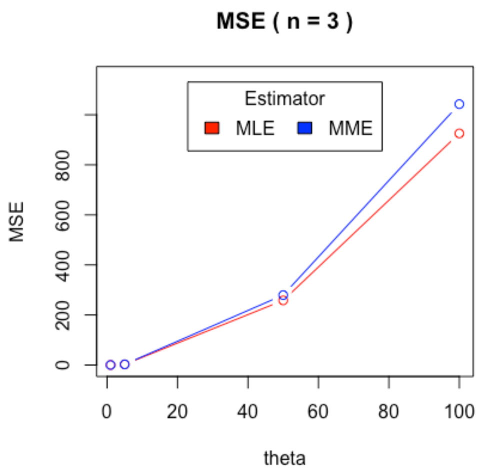
Solution:

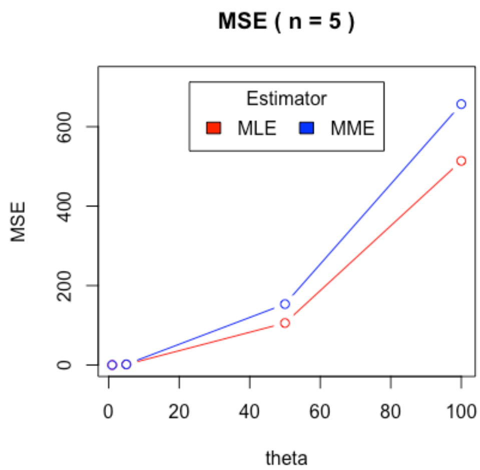


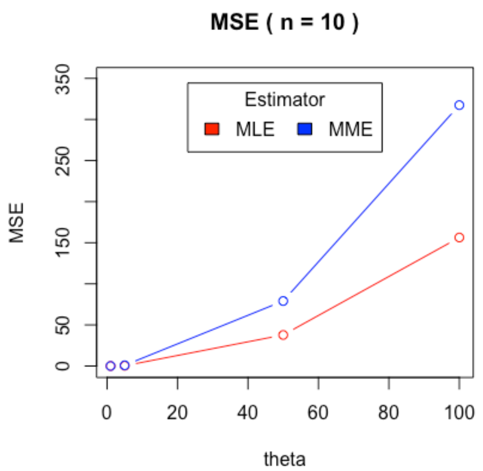
While θ stays and n increases, the MSEs of both MLE and MME decrease. When sample size n reach 30 which can be considered as large enough, the MSEs are close to 0. In most time, MLE < MME which means MLE is better than MME.

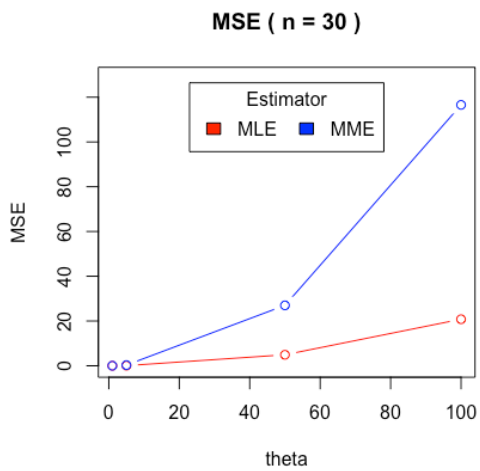












While n stays and θ increases, the MSEs of both MLE and MME increase. In most time, MLE

< MME which means MLE is better than MME. Since all populations share the same distribution, which is uniform, we can do further research about θ and MSE.

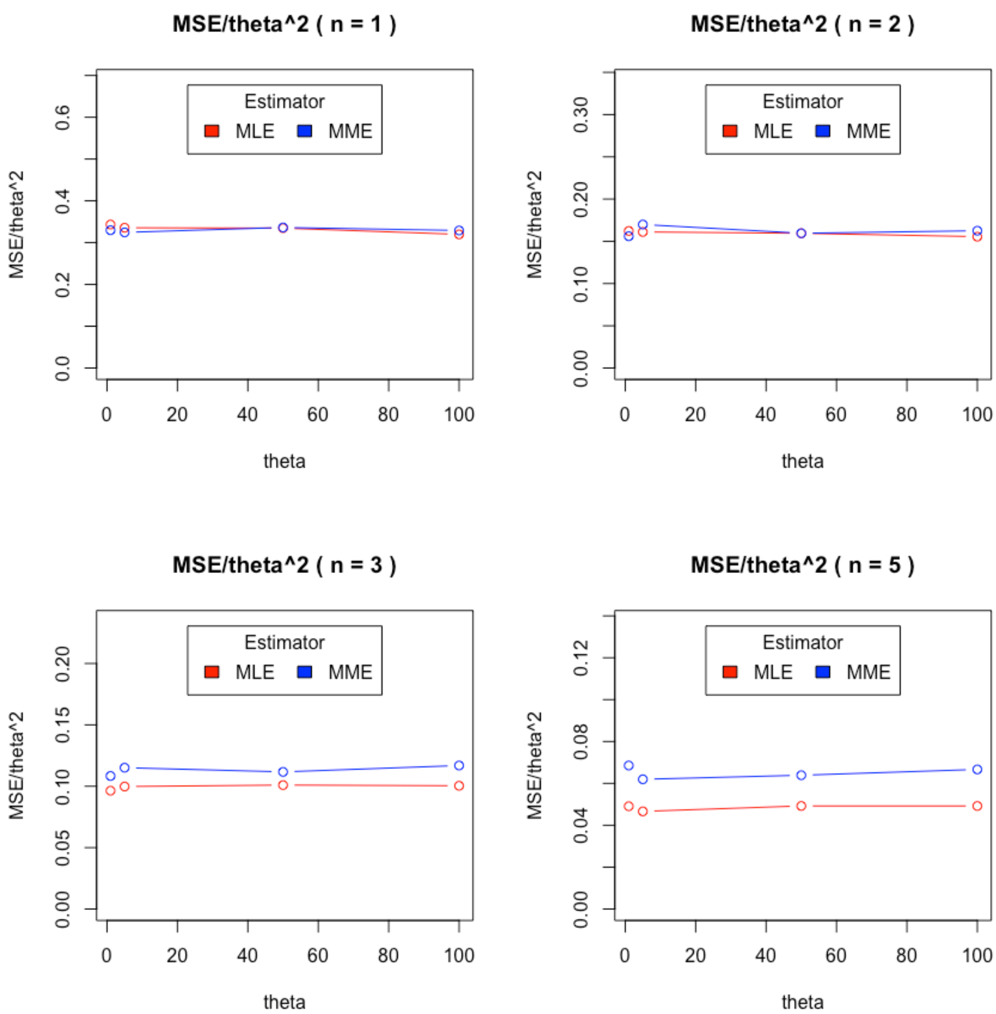
R code:

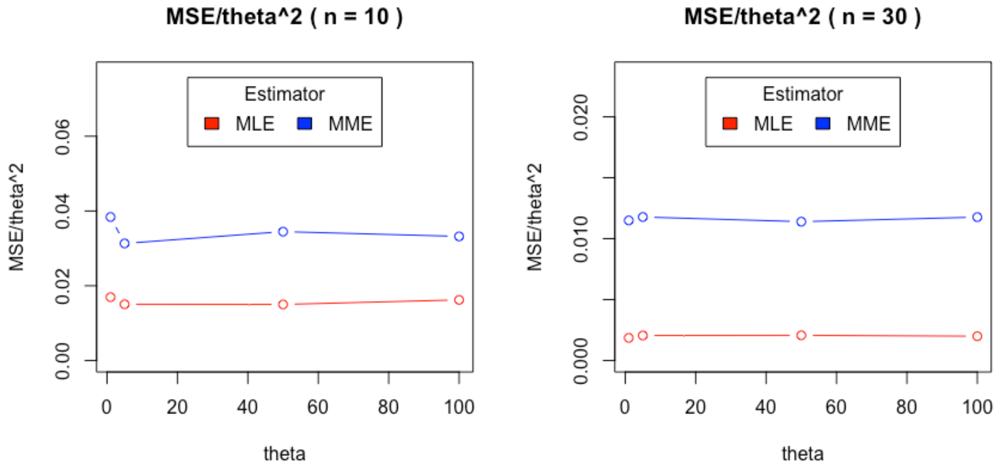
```r
# (d) Based on (c), which estimator is better? Does the answer depend on n or θ?
# Explain. Provide justification for all your conclusions.

# about theta
for (i in seq_along(ns)){
  y1=res.mle[i,]/((thetas)^2)
  y2=res.mme[i,]/((thetas)^2)
  lim = c(0, max(max(y1), max(y2))*2)
  plot(x=thetas, y=y1, type="b", xlab="theta", ylab="MSE/theta^2", ylim=lim,
       col="red", main=paste("MSE/theta^2 ( n =",ns[i],")"))
  lines(x=thetas, y=y2, type="b", col="blue")
  legend("top", inset=.05, title="Estimator", c("MLE","MME"),
         fill=c("red", "blue"), horiz=TRUE)
}

```

Diagrams:





All six diagrams show almost horizontal line which means: MSE $\propto$ θ^2.

In short, MLE estimator is better than MME estimator and this relationship does not rely on n or θ. However, MSE itself is related to n and θ. When n increases, MSE will decreases. When θ increases, MSE will increases. To be more precise, we have MSE $\propto$ θ^2.

Question2:

(a) Derive an expression for maximum likelihood estimator of θ.

Solution:

Step 1: $L(\theta) = \prod_{i=1}^{n} f(x_i) = \prod_{i=1}^{n} \left(\frac{\theta}{x_i^{\theta+1}}\right) = \frac{\theta^n}{\prod_{i=1}^{n} x_i^{\theta+1}}$

Then we can get $log(L(\theta)) = log\left(\frac{\theta^n}{\prod_{i=1}^{n} x_i^{\theta+1}}\right)$

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

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

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

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

Step 2: Differentiate $log(L(\theta))$ with respect θ, and set it to 0.

$$\frac{d\,log(L(\theta))}{d\theta} = \frac{n}{\theta} - \sum_{i=1}^{n} log(x_i) = 0$$

Then we can get $\hat{\theta}_{MLE} = \frac{n}{\sum_{i=1}^{n} log(x_i)}$

(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:

Take the values into $\hat{\theta}_{MLE} = \frac{n}{\sum_{i=1}^{n} log(x_i)}$ we can get:

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

$$= \frac{5}{log(5185263.52)}$$

$$= \frac{5}{15.4613}$$

$$= 0.32339$$

(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:

The R code is as following:

```
# (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?

# we will work with the lifetime data x in the question
x <- c(21.72, 14.65, 50.42, 28.78, 11.23)

# Negative of log-likelihood function

neg.loglik.fun <- function(par, dat){
  result <- length(dat)*log(par)-(par+1)*sum(log(dat))
  return(-result)
}

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

ml.est <- optim(par=0.1, fn=neg.loglik.fun, method="L-BFGS-B", lower=0.1,
                hessian=TRUE, dat=x)
print(ml.est)

```

The result is:

```
> print(ml.est)
$par
[1] 0.3233885

$value
[1] 26.10585

$counts
function gradient
       9        9

$convergence
[1] 0

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

$hessian
         [,1]
[1,] 47.81116
```

The numerically estimated result 0.3233885 matches the result we get in question(b).

(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 θ. Are these approximations going to be good? Justify your answer.

Solution:

In order to find the approximate 95% confidence interval for θ, we can use the following formula:

$$\text{CI:} \quad [\hat{\theta}_{MLE} - Z_{\frac{\alpha}{2}} * \sqrt{\hat{I}^{-1}},\ \hat{\theta}_{MLE} + Z_{\frac{\alpha}{2}} * \sqrt{\hat{I}^{-1}}]$$

$$\alpha = 1 - 0.95 \quad \textit{then} \ \ 1 - \frac{\alpha}{2} = 1 - 0.025 = 0.975$$

And $\hat{I}$ is the hessian function we can get from (C)

Then we can use R (below) to get the approximate 95% confidence interval for θ is

[0.03993389, 0.60684301]

This means that if we repeat a large number of times to estimate θ from randomly selected samples from the population, then the true estimate value lies in the interval [0.03993389, 0.60684301] 95% of the times.

However, this population is non-normal. So, we use the Central Limit Theory to estimate θ with $Z_{\frac{\alpha}{2}}$, but the sample size is just 5 which is not large. Thus, the confidence interval may be not very accurate.

The R code is:

```r
# (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 θ. Are these approximations going to be good? Justify
# your answer.

# standard errors
SE <- sqrt(solve(ml.est$hessian))[1]# announce [1] here is to avoid warning in R

# The confidence interval
ml.est$par + c(-1,1)*SE*qnorm(0.975)
```

The result is:

```
> # The confidence interval
> ml.est$par + c(-1,1)*SE*qnorm(0.975)
[1] 0.03993389 0.60684301
>
```