CS 6313 Statistical Methods in Data Science
Mini Project 3

Gaurav Joshi

March 19, 2022

Question 1

(a)

In order to calculate mean-squared error using Monte Carlo simulation, we use the population parameter θ and the sample values.

- Set population parameter
- Simulate sample values which allow calculation of estimator value.
- Compute Mean Squared Error = Estimator value – population parameter

(b)

The `generate_mom_mle` function returns a vector of variables `maximum_likelihood_est` and the `method_of_moments`. The variables are calculated as follows.

- `generate_uniform_dist:`
  Use the built-in `run_if` function that generates uniform dist from 0 to θ
- `maximum_likelihood_est:`
  Maximum likelihood is the maximum value of the uniform distribution.
- `method_of_moments:`
  Method of moments is the mean of uniform distribution times 2.

R Code:

```r
generate_mom_mle <- function(n, theta) {
  # Create uniform distribution between 0 to theta
  generate_uniform_dist <- runif(n, min=0, max=theta)

  # Calculate maximum likelihood estimator
  maximum_likelihood_est <- max(generate_uniform_dist)
  method_of_moments <- 2 * mean(generate_uniform_dist)

  return(c(method_of_moments, maximum_likelihood_est))
}

mse_est <- function(n, theta) {
  est <- replicate(1000, mse(n, theta))

  # calculate mean squared err
  est <- (est - theta)^2

  methodOfMoments <- est[c(FALSE, TRUE)]
  maxLikelihoodEst <- est[c(TRUE, FALSE)]

  # store mean of mom and mle
  mean_of_mom <- mean(methodOfMoments)
  mean_of_mle <- mean(maxLikelihoodEst)

  return(c(mean_of_mom, mean_of_mle))
}

mean_sq_err <- function(mse_est, n, theta) {
  return(mse_est(n, theta))
}
```

The `mse_est` function calculates mean squared error for $N = 1000$. The methodOfMoments and maxlikelihoodEst are first computed and utilized to give mean squared error by computing $(\hat{\theta} - \theta)^2$. We return the mean of methodOfMoments and maxlikelihoodEst.

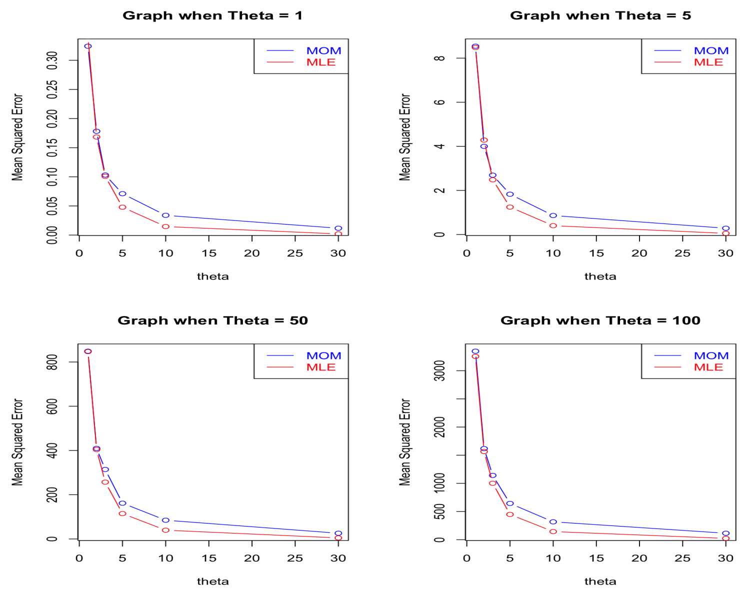
The `mean_sq_err` function takes as input the function `mse_est` and its two parameters n, θ and returns its result.

Screenshot of the Output:

```
> mean_sq_err(mse_est, 1, 1)
[1] 0.3430506 0.3269060
```

(c)

Graphs when N is constant for different values of θ



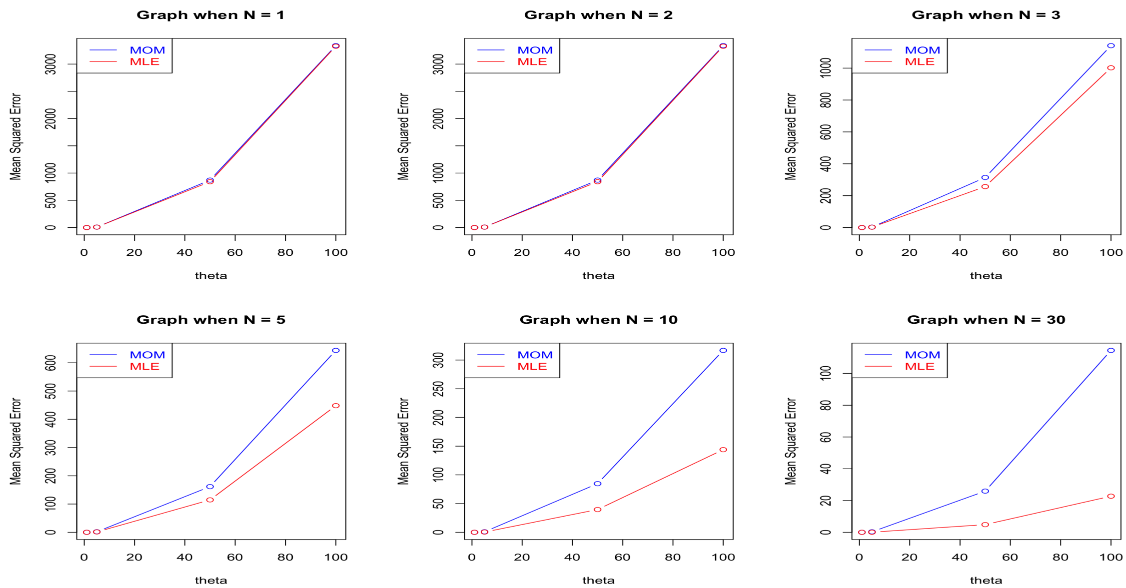
R Code:

```r
# When N = 1, calculate the mean squared error for different values of theta
one_one <- mean_sq_err(mse_est, 1, 1)
two_one <- mean_sq_err(mse_est, 2, 1)
three_one <- mean_sq_err(mse_est, 3, 1)
five_one <- mean_sq_err(mse_est, 5, 100)
ten_one <- mean_sq_err(mse_est, 10, 100)
thirty_one <- mean_sq_err(mse_est, 30, 100)
# The above values will be re-computed for theta = 1, 5, 50 and 100
# For example for theta = 5, calculate one_five, two_five ... thirty_five

# create vectors from input, mle and mse
input_vals_for_N <- c(1, 2, 3, 5, 10, 30)
mom_vals <- c(one_one[1], two_one[1], three_one[1],
              five_one[1], ten_one[1], thirty_one[1])
mle_vals <- c(one_one[2], two_one[2], three_one[2],
              five_one[2], ten_one[2], thirty_one[2])
textColor <- c("blue", "red")

# plot graph mse when theta = 1, for N = {1, 2, 3, 5, 10, 30}
plot(input_vals_for_N, mom_vals, main = "Graph when Theta = 1", type="b",
     xlab="theta", ylab="Mean Squared Error", col="blue")
lines(input_vals_for_N, mle_vals, type="b", col="red")
legend("topright", legend = c("MOM", "MLE"), col = textColor,
        text.col = textColor, ncol = 1, lty = 1)
```

Graphs when θ is constant for different values on N:



R Code:

```
# When N = 1, calculate the mean squared error for different values of theta
one_one <- mean_sq_err(mse_est, 1, 1)
one_five <- mean_sq_err(mse_est, 1, 5)
one_fifty <- mean_sq_err(mse_est, 1, 50)
one_hundr <- mean_sq_err(mse_est, 1, 100)
# The above values will be re-computed for N = 2, 3, 5, 10 and 30
# Substitute value of N int mse_est function

# When N = 2, one_one <- mean_sq_err(mse_est, 2, 1) and so on

# create vectors from input, mle and mse
input_vals_for_theta <- c(1, 5, 50, 100)
mse_vals <- c(one_one[1], one_five[1], one_fifty[1], one_hundr[1])
mle_vals <- c(one_one[2], one_five[2], one_fifty[2], one_hundr[2])
textColor <- c("blue", "red")

# plot graph mse when N = 1, for theta = {1, 5, 50, 100}
plot(input_vals_for_theta, mse_vals, main = "Graph when N = 30", type="b",
     xlab="theta", ylab="Mean Squared Error", col="blue")
lines(input_vals_for_theta, mle_vals, type="b", col="red")
legend("topleft", legend = c("MOM", "MLE"), col = textColor,
       text.col = textColor, ncol = 1, lty = 1)
```

Note: As mentioned in the code, these values will be recomputed. The R code for both the types of graphs is given only for N = 1 and $\theta = 1$. While the core logic will remain the same, these values will be recomputed as mentioned in the code comments.

(d)

For the graphs when θ is constant, we can clearly see that as the value of N increases, the difference between Mean Squared Error for Maximum Likelihood Estimator and Method of Moments becomes substantial.

Intuitively, it makes sense for large errors when N = 1, 2 and 3.

But as N becomes bigger, it is clear that MLE is much more accurate than MOM and hence a better choice for large values of N.

For the graphs when N is fixed for different values of θ both the Maximum Likelihood Estimator and Method of Moments are nearly inseparable.

From these two types of graphs, it is clear that the Mean Squared Error depends on N not θ and MLE is a better choice for larger values of N.

Question 2

(a)

For the given probability density function, let the likelihood function be L(θ).

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

Taking log on both sides,

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

After differentiating the equation, the results can be set to 0. This gives the equation,

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

$$\frac{n}{\theta} = \sum_{i=1} log(x_i)$$

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

(b)

Using the equation derived in 2a, we can calculate maximum likelihood estimator of θ given by $\hat{\theta}$. We know that,

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

Since we are given the 5 values of x, we set n = 5 and compute the log values of $x_1, x_2 ... x_5$ by plugging in the given values.

$$\hat{\theta}_{MLE} = \frac{5}{log(x_1) + log(x_2) + log(x_3) + log(x_4) + log(x_5)}$$

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

$$= \frac{5}{3.079 + 2.684 + 3.92 + 3.36 + 2.419}$$

$$= \frac{5}{15.46}$$

$$= 0.3234$$

Therefore, $\hat{\theta}_{MLE} = 0.3234$

(c)

The `optim` function can be used to give the maximum likelihood estimator. The `log_likelihood` function emits a negative val because the default output minimizes the function so negating it maximizes the function.

This function becomes the input to the `optim` function.

R Code:

```r
log_max_likelihood <- function(theta, dat) {
  res <- length(dat) * log(theta) - (theta + 1) * sum(log(dat))
  return (-res)
}

input <- c(21.42, 14.65, 50.42, 28.78, 11.23)

max_likelihood_est <- optim(par=1, fn = log_max_likelihood,
                            method = "L-BFGS-B", hessian = TRUE,
                            lower = 0.01, dat = input)
```

Executing the `max_likelihood_est` function, we get the following result for the par (which is the MLE for $\hat{\theta}$) which is very close to the calculated $\hat{\theta}_{MLE}$

Screenshot of the Output

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

(d)

Standard error, SE is given by:

$$SE(\hat{\theta}) = \sqrt{\frac{1}{\hat{\Gamma}}}$$

Here $\hat{\Gamma}$ represents hessian function.

The hessian function can be used to calculate SE. The standard error along with the output of MLE from 2c is used to compute CI (confidence interval).

$$1 - \alpha = 0.95$$

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

Thus, the confidence interval would be $\overline{\sigma} \pm \overline{Z}_{\frac{\alpha}{2}} * SE(\hat{\theta})$

Using the `qnorm` function, we calculate the CI. Since $\frac{\alpha}{2} = 0.025$, $1 - \frac{\alpha}{2} = 0.975$. We use this value to compute CI.

R Code:

```r
hessian_val <- max_likelihood_est$hessian[1]
standard_error <- (1/hessian_val)^0.5

#confidence interval
par_of_mle <- max_likelihood_est$par
ci <- par_of_mle + c(-1, 1) * as.vector(standard_error) * qnorm(0.975)
```

Standard Error

```
> standard_error
[1] 0.1447525
```

Confidence Interval

```
> ci
[1] 0.03996984 0.60738939
```

The derived Standard Error is: 0.1447525. The derived CI is: (0.03996984, 0.60738939)

From the results it can be said that the true estimates lie between these intervals 95% of the time.