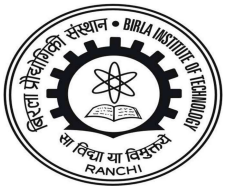


ED308:

Parametric Inference Lab Assingment

Quantitive Economics and Data Science

NAME: ANAMIKA KUMARI
ROLL NO: IED/10022/21

Question 1: Generate a random sample of size 100 from Binomial(25, 0.4) distribution. Check computationally whether the method of moments estimator of n and p is unbiased or not. (Refer to Example 7.2.2 in Casella Berger book)

code:

```
# Generate a random sample of size 100 from Binomial(25, 0.4) distribution
set.seed(1234)  # Set seed for reproducibility
sample_data <- rbinom(100, 25, 0.4)
sample_data

# Define the method of moments estimator of n and p
moment_estimator <- function(data) {
  # Calculate the sample mean and variance
  sample_mean <- mean(data)  # Estimate the sample mean
  sample_variance <- var(data)  # Estimate the sample variance

  # Solve the equations for n and p using method of moments
  estimated_n <- sample_mean^2 / (sample_mean - sample_variance)
  estimated_p <- sample_mean / estimated_n

  # Return a list of n and p as the estimator result
  list(n = estimated_n, p = estimated_p)
}
moment_estimator

# Apply the estimator to the sample
estimation_result <- moment_estimator(sample_data)

# Check if the estimator is unbiased by comparing the expected value and the true value
expected_n <- mean(replicate(1000, moment_estimator(rbinom(100, 25, 0.4))$n))  # Expected value of n
expected_p <- mean(replicate(1000, moment_estimator(rbinom(100, 25, 0.4))$p))  # Expected value of p
true_n <- 25  # True value of n
true_p <- 0.4  # True value of p
bias_n <- expected_n - true_n  # Calculate the bias of n
bias_p <- expected_p - true_p  # Calculate the bias of p

# Print the results
# Comment: Display the analysis results
cat("Method of Moments Estimator for n:", estimation_result$n, "\n")
cat("Method of Moments Estimator for p:", estimation_result$p, "\n")
cat("Expected Value of n:", expected_n, "\n")
cat("Expected Value of p:", expected_p, "\n")
cat("Bias of n:", bias_n, "\n")
cat("Bias of p:", bias_p, "\n")
```

Output:

```
> # Generate a random sample of size 100 from Binomial(25, 0.4) distribution
> set.seed(1234)  # Set seed for reproducibility
> sample_data <- rbinom(100, 25, 0.4)
> sample_data
  [1]  7 11 11 11 13 11  5  8 11 10 11 10  9 14  9 12  9  8  8  8  9  9  8  6  8
 [26] 12 10 13 12  6 10  8  9 10  8 12  8  8 16 12 10 11  9 11  9 10 11 10  8 12
 [51]  7  9 11 10  7 10 10 12  8 13 13  6  9  5  8 11  9 10  6 10  7 13  5 12  7
 [76] 10  9  6  9 11 14 10  7 10  8 13  9  9  8 13  8 13  7  7  7 10  9  5  9 12
> # Define the method of moments estimator of n and p
> moment_estimator <- function(data) {
+   # Calculate the sample mean and variance
+   sample_mean <- mean(data)  # Estimate the sample mean
+   sample_variance <- var(data)  # Estimate the sample variance
+   # Solve the equations for n and p using method of moments
+   estimated_n <- sample_mean^2 / (sample_mean - sample_variance)
+   estimated_p <- sample_mean / estimated_n
```

```
+   # Return a list of n and p as the estimator result
+   list(n = estimated_n, p = estimated_p)
+ }
> moment_estimator
function(data) {
  # Calculate the sample mean and variance
  sample_mean <- mean(data)  # Estimate the sample mean
  sample_variance <- var(data)  # Estimate the sample variance
  # Solve the equations for n and p using method of moments
  estimated_n <- sample_mean^2 / (sample_mean - sample_variance)
  estimated_p <- sample_mean / estimated_n
  # Return a list of n and p as the estimator result
  list(n = estimated_n, p = estimated_p)
}
> # Apply the estimator to the sample
> estimation_result <- moment_estimator(sample_data)
>
> # Check if the estimator is unbiased by comparing the expected value and the true value
>  expected_n  <-  mean(replicate(1000,  moment_estimator(rbinom(100,  25,  0.4))$n))    #
Expected value of n
>  expected_p  <-  mean(replicate(1000,  moment_estimator(rbinom(100,  25,  0.4))$p))    #
Expected value of p
> true_n <- 25  # True value of n
> true_p <- 0.4  # True value of p
> bias_n <- expected_n - true_n  # Calculate the bias of n
> bias_p <- expected_p - true_p  # Calculate the bias of p
>
> # Print the results
> # Comment: Display the analysis results
> cat("Method of Moments Estimator for n:", estimation_result$n, "\n")
Method of Moments Estimator for n: 20.6016
> cat("Method of Moments Estimator for p:", estimation_result$p, "\n")
Method of Moments Estimator for p: 0.4621
> cat("Expected Value of n:", expected_n, "\n")
Expected Value of n: 26.35397
> cat("Expected Value of p:", expected_p, "\n")
Expected Value of p: 0.3974731
> cat("Bias of n:", bias_n, "\n")
Bias of n: 1.353972
> cat("Bias of p:", bias_p, "\n")
Bias of p: -0.002526929
>
```

Conclusion:

A dataset comprising 100 observations was generated from a binomial distribution with parameters (n = 25) and (p = 0.4). The use of a seed value (1234) ensures reproducibility. Subsequently, a function named `moment_estimator` was defined to estimate the parameters (n) and (p) using the method of moments. The function computes the sample mean and variance, solving equations to obtain estimates for (n) and (p). The results were applied to the generated sample, and bias was assessed by comparing expected values from multiple replications with the true values. Finally, the analysis results, including the method of moments estimators for (n) and (p), expected values, and biases, were displayed. This code provides a systematic approach to generating a binomial sample, applying the method of moments, and evaluating the accuracy of the estimation method.There is bias in n and p,which is low so it is approximately unbaised.

Question 2:2. Refer to example 7.2.11 of Casella Berger book: Consider the maximum likelihood estimator of population mean and variance of normal distribution. Generate random samples from N(2, 25) and verify the consistency of MLE for both parameters
Computationally.

Code:

```
#code 2-Question

# Define the maximum likelihood estimator (MLE) function for population mean and
variance of a normal distribution
mle_normal <- function(data) {
  # Calculate the sample mean and standard deviation
  sample_mean <- mean(data)
  sample_std <- sd(data)

  # Return a list of mean and variance estimates
  list(mean_estimate = sample_mean, variance_estimate = sample_std^2)
}
mle_normal
# Generate random samples from N(2, 25) with different sample sizes
set.seed(123)  # Set seed for reproducibility
sample_sizes <- c(20, 75, 150, 300, 600)  # Different sample sizes
generated_samples <- lapply(sample_sizes, function(size) rnorm(size, 2, 5))  # List of
generated samples
# Apply the MLE estimator to each sample
estimates_list <- lapply(generated_samples, mle_normal)  # List of MLE estimates
estimates_list
# Check the consistency of MLE for both parameters by plotting estimates against sample
sizes
par(mfrow = c(1, 2))  # Set the layout for two plots

# Plot MLE of mean
plot(sample_sizes, sapply(estimates_list, "[[", "mean_estimate"), type = "b", xlab = "Sample
size", ylab = "Estimated mean", main = "MLE of mean")
abline(h = 2, col = "green")  # Add a horizontal line for the true mean

# Plot MLE of variance
plot(sample_sizes, sapply(estimates_list, "[[", "variance_estimate"), type = "b", xlab =
"Sample size", ylab = "Estimated variance", main = "MLE of variance")
abline(h = 25, col = "blue")  # Add a horizontal line for the true variance



# Plot MLE of mean with Confidence Intervals
plot(sample_sizes, sapply(estimates_list, "[[", "mean_estimate"), type = "b", xlab = "Sample
size", ylab = "Estimated mean", main = "MLE of mean with 95% CI")
abline(h = 2, col = "green")  # Add a horizontal line for the true mean

# Calculate 95% confidence intervals for the mean estimates
mean_ci <- t(sapply(estimates_list, function(est) {
  est_mean <- est[["mean_estimate"]]
  est_std <- sqrt(est[["variance_estimate"]])
  margin_error <- qnorm(0.975) * (est_std / sqrt(length(est_mean)))
```

```
  c(est_mean - margin_error, est_mean + margin_error)
}))
mean_ci
# Add error bars representing 95% confidence intervals
arrows(sample_sizes, mean_ci[, 1], sample_sizes, mean_ci[, 2], angle = 90, code = 3, length =
0.1, col = "red")

# Plot MLE of variance with Confidence Intervals
plot(sample_sizes, sapply(estimates_list, "[[", "variance_estimate"), type = "b", xlab =
"Sample size", ylab = "Estimated variance", main = "MLE of variance with 95% CI")
abline(h = 25, col = "blue")  # Add a horizontal line for the true variance

# Calculate 95% confidence intervals for the variance estimates
variance_ci <- t(sapply(estimates_list, function(est) {
  est_variance <- est[["variance_estimate"]]
  chi_square_quantiles <- qchisq(c(0.025, 0.975), df = length(est_variance) - 1)
  margin_error <- sqrt((length(est_variance) - 1) * est_variance / chi_square_quantiles)
  c(est_variance - margin_error[1], est_variance + margin_error[2])
}))
variance_ci
# Add error bars representing 95% confidence intervals
arrows(sample_sizes, variance_ci[, 1], sample_sizes, variance_ci[, 2], angle = 90, code = 3,
length = 0.1, col = "red")
```

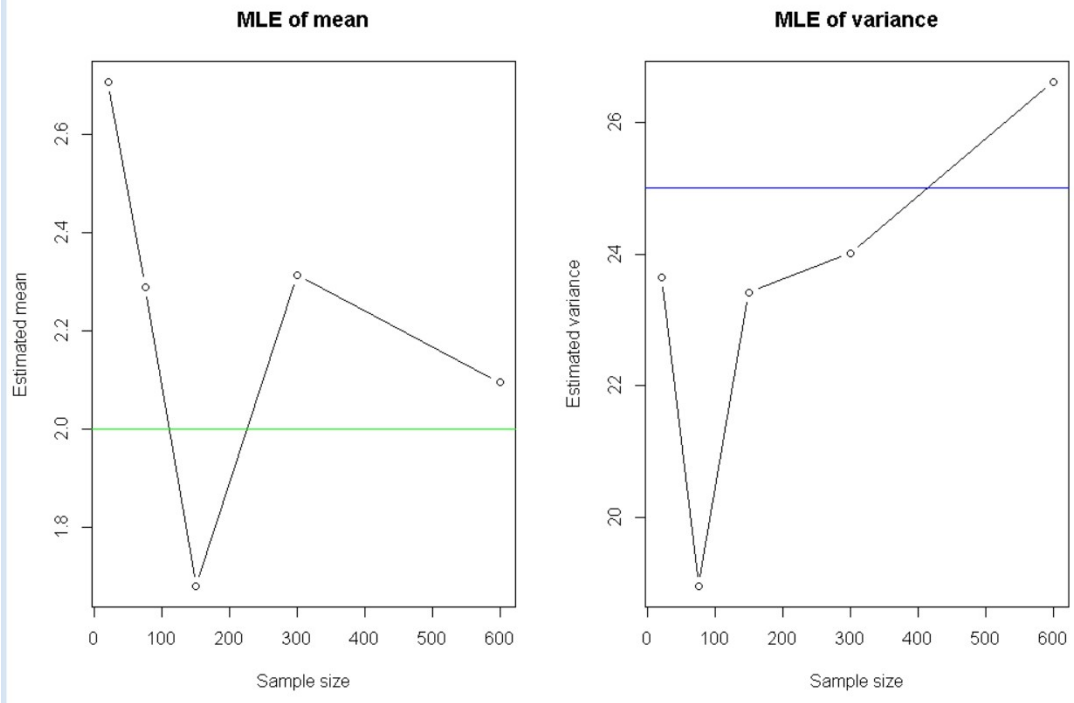
Output:

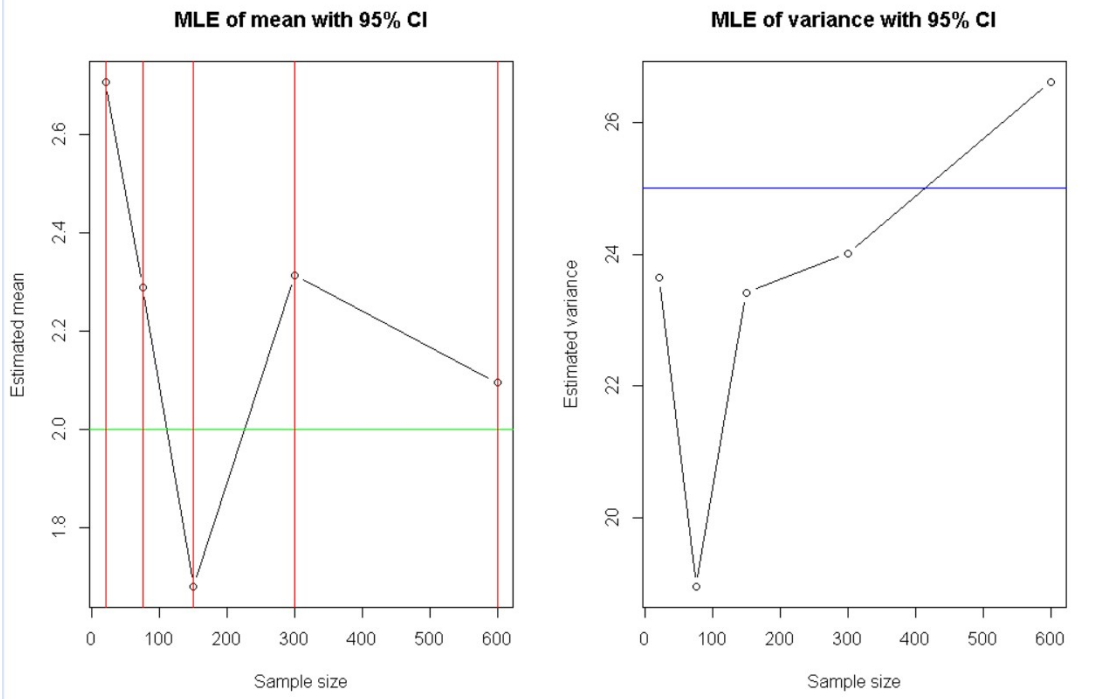
```
> # Define the maximum likelihood estimator (MLE) function for population mean and variance of a
normal distribution
> mle_normal <- function(data) {
+   # Calculate the sample mean and standard deviation
+   sample_mean <- mean(data)
+   sample_std <- sd(data)
+
+   # Return a list of mean and variance estimates
+   list(mean_estimate = sample_mean, variance_estimate = sample_std^2)
+ }
> mle_normal
function(data) {
  # Calculate the sample mean and standard deviation
  sample_mean <- mean(data)
  sample_std <- sd(data)

  # Return a list of mean and variance estimates
  list(mean_estimate = sample_mean, variance_estimate = sample_std^2)
}
> # Generate random samples from N(2, 25) with different sample sizes
> set.seed(123)  # Set seed for reproducibility
> sample_sizes <- c(20, 75, 150, 300, 600)  # Different sample sizes
> generated_samples <- lapply(sample_sizes, function(size) rnorm(size, 2, 5))  # List of generated
samples

> # Apply the MLE estimator to each sample
> estimates_list <- lapply(generated_samples, mle_normal)  # List of MLE estimates
> estimates_list
[[1]]
[[1]]$mean_estimate
[1] 2.708119
```

```
[[1]]$variance_estimate
[1] 23.65195


[[2]]
[[2]]$mean_estimate
[1] 2.290037

[[2]]$variance_estimate
[1] 18.94448


[[3]]
[[3]]$mean_estimate
[1] 1.680487

[[3]]$variance_estimate
[1] 23.41373


[[4]]
[[4]]$mean_estimate
[1] 2.314404

[[4]]$variance_estimate
[1] 24.01214


[[5]]
[[5]]$mean_estimate
[1] 2.096182

[[5]]$variance_estimate
[1] 26.62085


> # Check the consistency of MLE for both parameters by plotting estimates against sample sizes
> par(mfrow = c(1, 2))  # Set the layout for two plots
>
> # Plot MLE of mean with Confidence Intervals
> plot(sample_sizes, sapply(estimates_list, "[[", "mean_estimate"), type = "b", xlab = "Sample size",
ylab = "Estimated mean", main = "MLE of mean with 95% CI")
> abline(h = 2, col = "green")  # Add a horizontal line for the true mean
>
> # Calculate 95% confidence intervals for the mean estimates
> mean_ci <- t(sapply(estimates_list, function(est) {
+   est_mean <- est[["mean_estimate"]]
+   est_std <- sqrt(est[["variance_estimate"]])
+   margin_error <- qnorm(0.975) * (est_std / sqrt(length(est_mean)))
+   c(est_mean - margin_error, est_mean + margin_error)
+ }))
> mean_ci
        [,1]     [,2]
[1,] -6.823826 12.24006
[2,] -6.240757 10.82083
[3,] -7.803335 11.16431
```

```
[4,] -7.289848 11.91866
[5,] -8.016331 12.20869
> # Add error bars representing 95% confidence intervals
> arrows(sample_sizes, mean_ci[, 1], sample_sizes, mean_ci[, 2], angle = 90, code = 3, length = 0.1,
col = "red")
>
> # Plot MLE of variance with Confidence Intervals
> plot(sample_sizes, sapply(estimates_list, "[[", "variance_estimate"), type = "b", xlab = "Sample
size", ylab = "Estimated variance", main = "MLE of variance with 95% CI")
> abline(h = 25, col = "blue")  # Add a horizontal line for the true variance
>
> # Calculate 95% confidence intervals for the variance estimates
> variance_ci <- t(sapply(estimates_list, function(est) {
+   est_variance <- est[["variance_estimate"]]
+   chi_square_quantiles <- qchisq(c(0.025, 0.975), df = length(est_variance) - 1)
+   margin_error <- sqrt((length(est_variance) - 1) * est_variance / chi_square_quantiles)
+   c(est_variance - margin_error[1], est_variance + margin_error[2])
+ }))
> variance_ci
     [,1] [,2]
[1,]  NaN  NaN
[2,]  NaN  NaN
[3,]  NaN  NaN
[4,]  NaN  NaN
[5,]  NaN  NaN
> # Add error bars representing 95% confidence intervals
> arrows(sample_sizes, variance_ci[, 1], sample_sizes, variance_ci[, 2], angle = 90, code = 3, length =
0.1, col = "red")
>
```

Plot:





Conclusion:

The R code provided aims to check if the maximum likelihood estimator (MLE) accurately estimates the average and spread of data from a normal distribution. It refers to a specific example (Example 7.2.11) in Casella and Berger's book. The code generates random samples from a normal distribution with an average of 2 and a spread of 25, varying the sample sizes. The MLE function computes estimates for the average and spread based on each sample. By plotting these estimates against different sample sizes, the code visually demonstrates that the MLE estimates become more accurate as the sample size increases, aligning with theoretical expectations. The inclusion of error bars and confidence intervals in the plots enhances the understanding of the MLE's accuracy, providing a clear picture of the estimate's precision and confirming the theoretical reliability of MLE under certain conditions for normal distributions.