Monte Carlo Simulation

Khalid

Monte Carlo Simulation

Pseudorandom

```r
# Normal Distribution
rnorm(100, mean=10,sd=4)
```

```
##   [1] 14.489275  8.934628 15.305237 14.183181  2.437171 11.611101  8.921117
##   [8] 13.324827  5.562299 11.540365  2.572792 19.827983 10.886026  6.664509
##  [15]  3.283109  9.503815 11.277550  6.908842 11.681220 13.100102 12.915570
##  [22] 17.794287 11.577196 16.019182  8.820506  8.450380  5.980694  6.079177
##  [29] 16.968824  7.389722 15.692854 11.359130 18.283430 14.614064 13.038349
##  [36] 14.713554 10.933216 14.651154  6.527663  9.952426 11.365262 11.190752
##  [43]  6.351921 16.818532  8.340246 11.703687  2.682426  9.501440  7.792164
##  [50]  5.224410  3.382744  7.188772 10.847969  4.086652 11.404529  4.088467
##  [57]  9.232098  9.058890  9.520688  6.394025  8.640311  8.638108  8.363253
##  [64] 10.904801 10.694641  9.665829  6.084653  9.195823 12.793533 11.800705
##  [71] 11.088143  4.008799 10.332644 10.627239 15.220193  6.934430  9.680247
##  [78] 16.012337 10.407743  7.497298 14.878978  7.555412  9.650342  8.572981
##  [85]  9.199300  9.610912 13.247081  6.890105 11.145709 13.555920  7.535279
##  [92]  5.988301 11.764094  9.685295  7.678374  8.204600  9.300023 11.543073
##  [99]  8.175769  4.752933
```

```r
# Binomial Distribution
rbinom(6,size=1,prob=0.5)
```

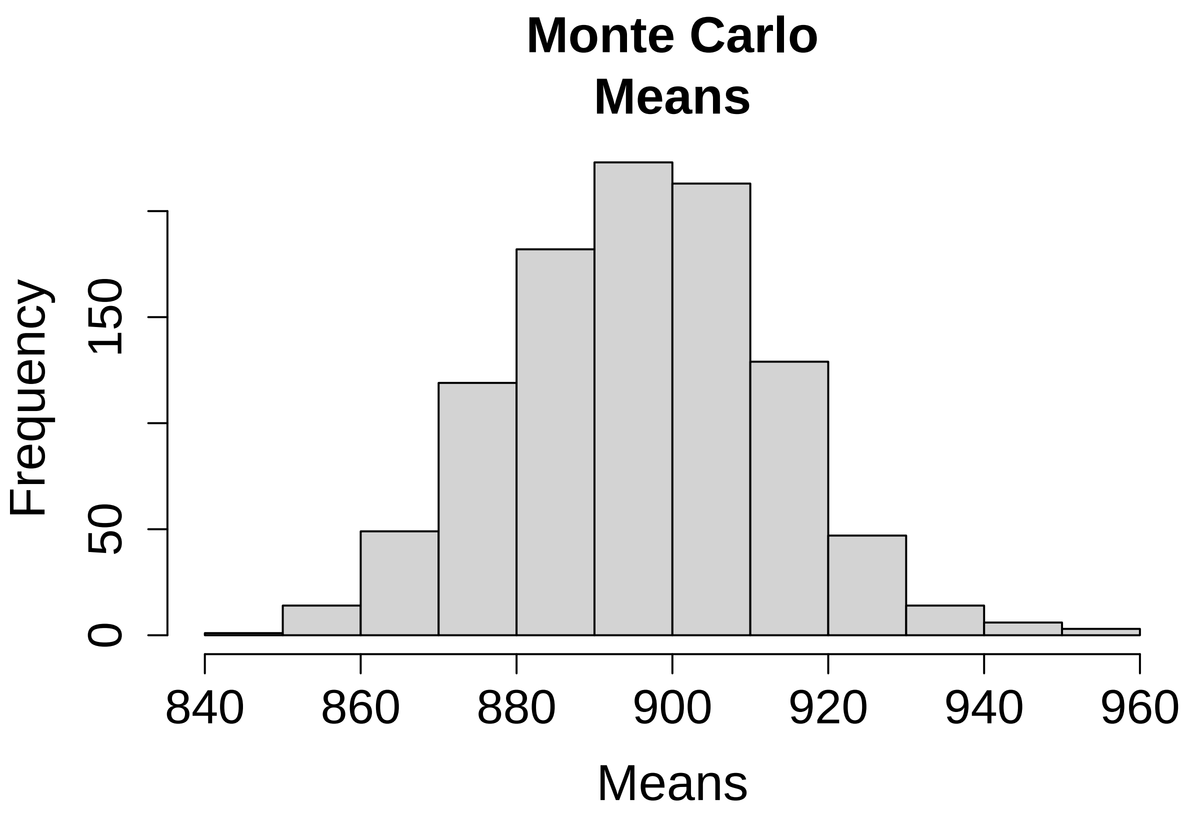
```
## [1] 1 0 1 0 0 0
```

```r
# Uniform Distribution
runif(50,min=0,max=100)
```

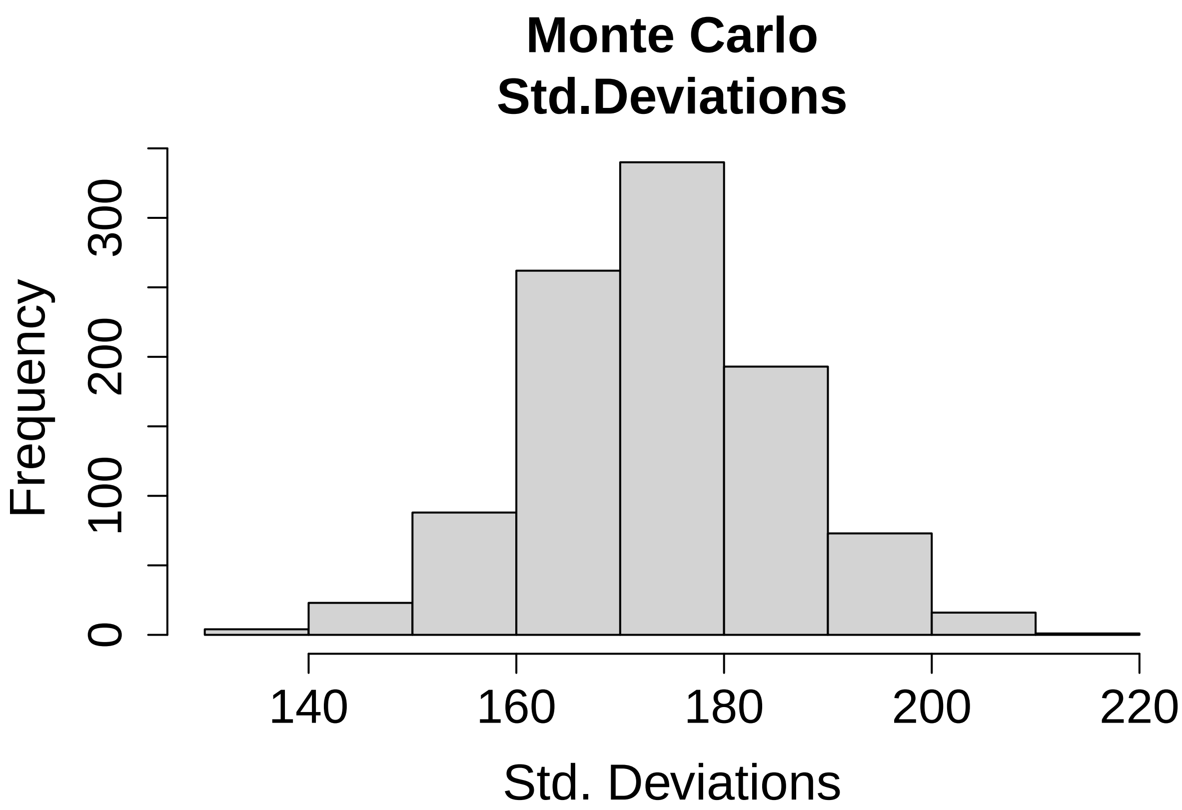
```
##  [1] 27.954971 58.879476 66.219484 71.596723 33.977561  8.080783 65.623898
##  [8] 93.418088 17.555550 29.686150 56.713923 25.365614 85.091491 27.601491
## [15] 21.380862 80.471349 68.311222 89.096457 75.407736 18.977294  2.823960
## [22] 70.410546 82.504433 82.000333 90.596377 49.982620 99.795988  3.064920
## [29] 31.743327 92.174078 33.049688 13.114782 23.793760 51.814249 26.795551
## [36] 37.045567 82.106863 26.982302  7.802446 46.522252 84.429265 51.513159
## [43] 38.651616 96.070249 84.006444 79.439332 94.963469 46.540003 66.605842
## [50] 64.416948
```

MCS Example

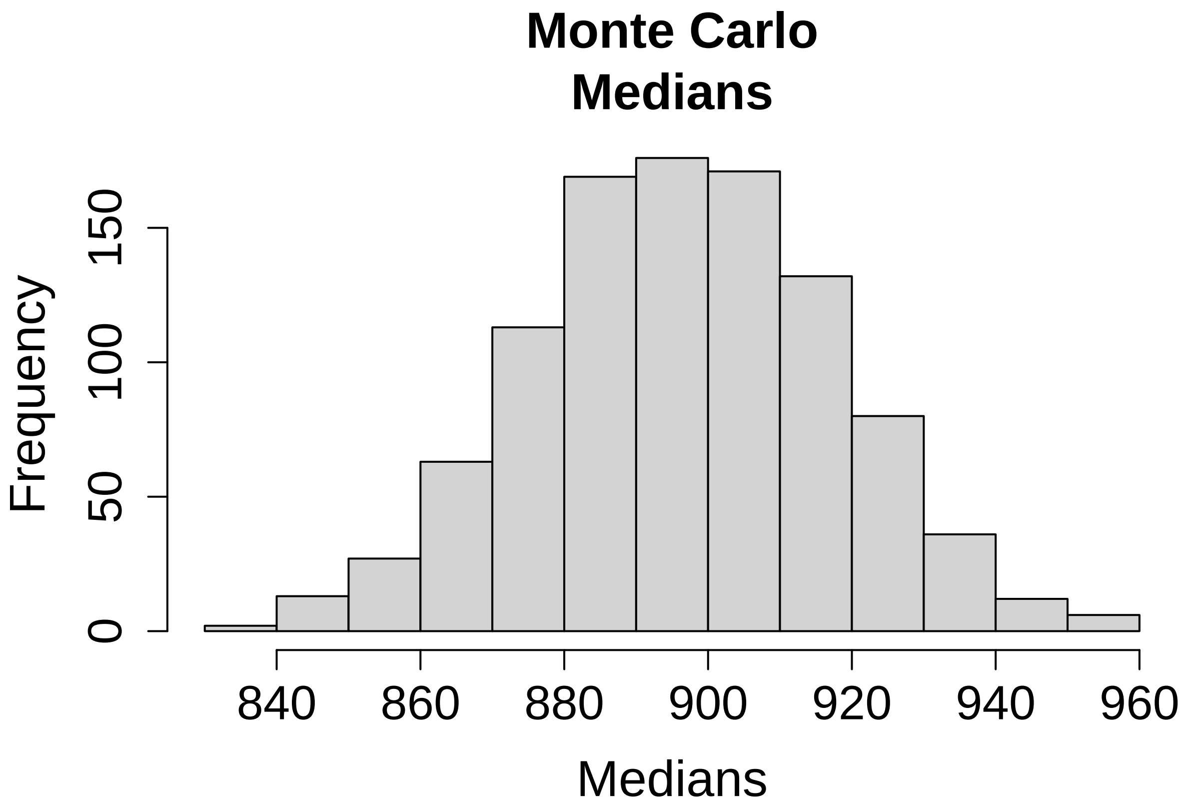
```r
#Data Genaration
m <- 1000
n <- 100
x <- matrix(rnorm((m*n),896,174), nrow=m)
means <- apply(x,MARGIN=1,FUN=mean)
sdevs <- apply(x,MARGIN=1,FUN=sd)
medians <- apply(x,MARGIN=1,FUN=median)
ranges <- apply(apply(x,MARGIN=1,FUN=range),MARGIN=2,FUN=diff)

# Visualisation

## Hist Mean
hist(means,xlab="Means",ylab="Frequency", main="Monte Carlo
Means",cex.axis=1.5,cex.lab=1.6,cex.main=1.6)
```



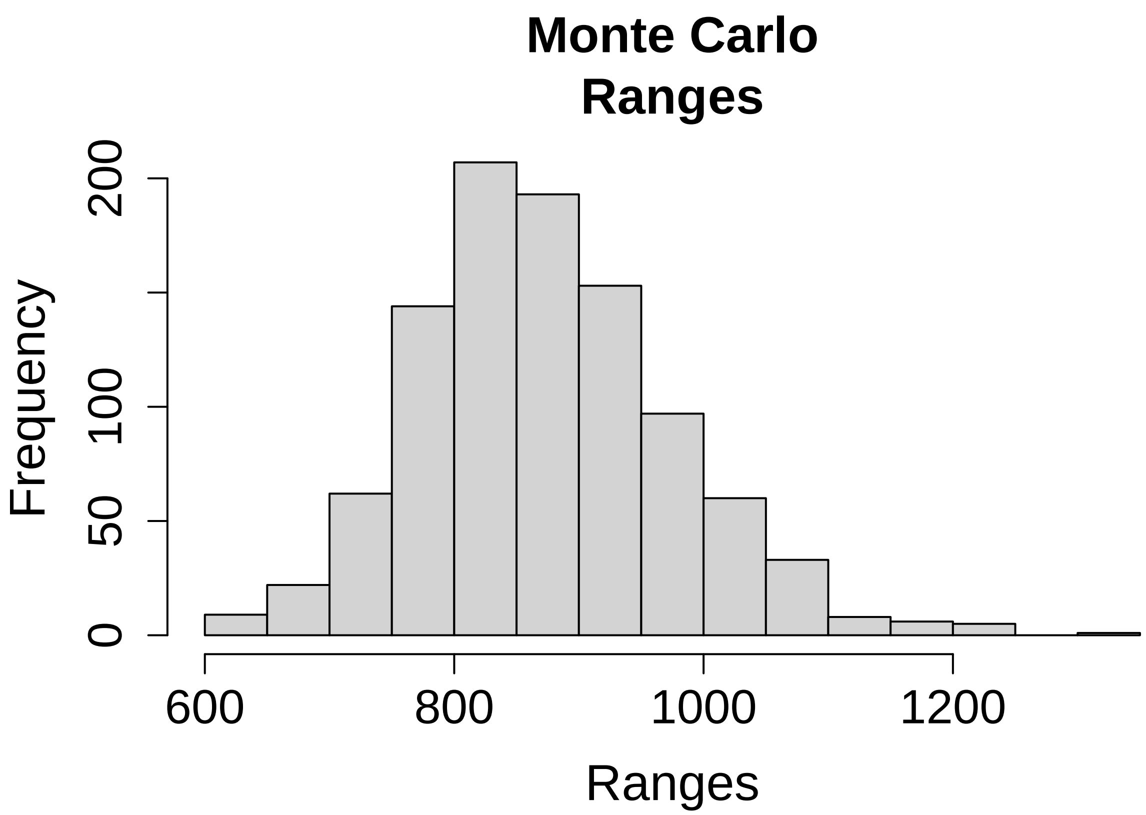

```r
## Hist Sdev
hist(sdevs,xlab="Std. Deviations",ylab= "Frequency",main="Monte Carlo
Std.Deviations",cex.axis=1.5,cex.lab=1.6,cex.main=1.6)
```




```r
## Hist Median
hist(medians,xlab="Medians",ylab="Frequency", main="Monte Carlo
Medians",cex.axis=1.5,cex.lab=1.6,cex.main=1.6)
```




```
##  Hist Range
hist(ranges,xlab="Ranges",ylab="Frequency", main="Monte Carlo
Ranges",cex.axis=1.5,cex.lab=1.6,cex.main=1.6)
```

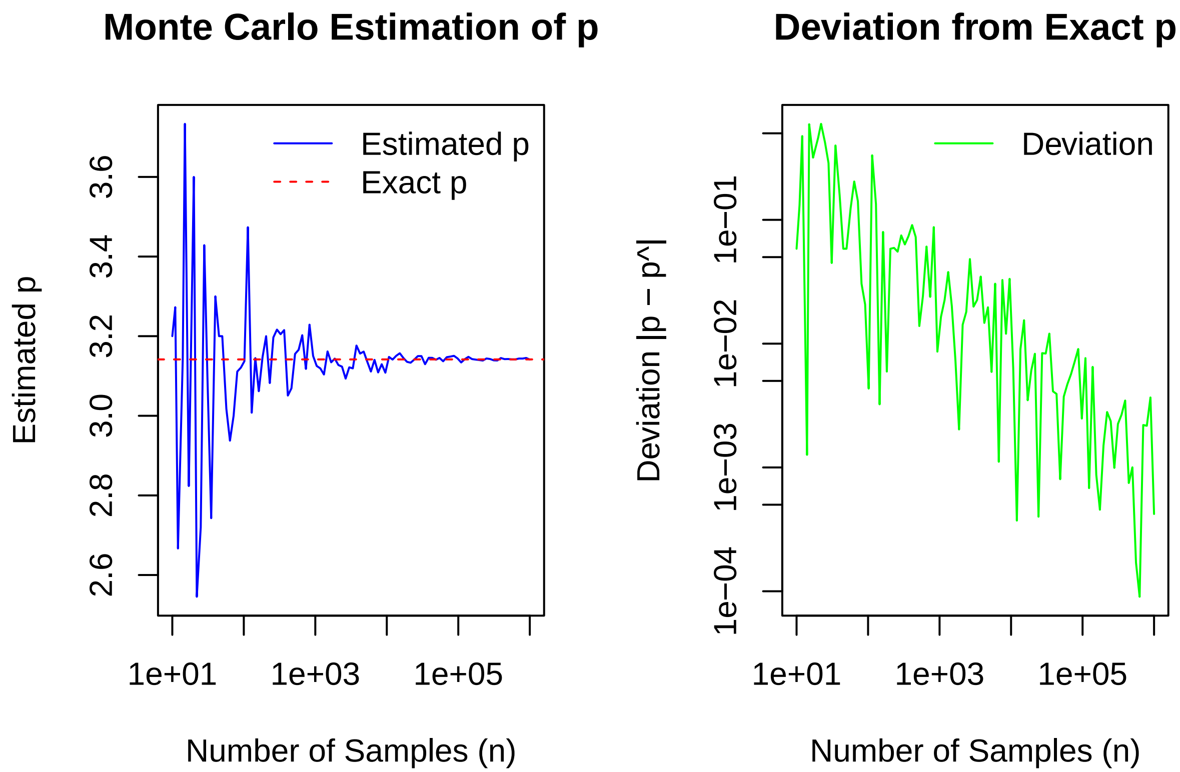


Exercise

Question 1

```
# Monte Carlo Simulation to Estimate
monte_carlo_pi <- function(n) {
  x <- runif(n, -1, 1) # produce x coordinate
  y <- runif(n, -1, 1) # produce y coordinate
  inside_circle <- (x^2 + y^2) <= 1  # Check if points fall inside the unit circle
```

```r
  pi_estimate <- 4 * sum(inside_circle) / n  # Estimate   using the ratio
  return(pi_estimate)
}

# Parameters for the simulation
n_values <- as.integer(10^seq(1, 6, length.out = 100))  # Vary n from 10 to 1,000,000
pi_exact <- pi  # Exact value of
estimates <- numeric(length(n_values))  # Store estimates of
deviations <- numeric(length(n_values))  # Store deviations from exact value

# Perform Monte Carlo simulation for different n values
for (i in seq_along(n_values)) {
  n <- n_values[i]
  estimates[i] <- monte_carlo_pi(n)
  deviations[i] <- abs(pi_exact - estimates[i])
}

# Plotting
par(mfrow = c(1, 2))  # Create a 1x2 plot layout

# Plot   estimates
plot(n_values, estimates, type = "l", log = "x", col = "blue",
     xlab = "Number of Samples (n)", ylab = "Estimated  ",
     main = "Monte Carlo Estimation of  ")
abline(h = pi_exact, col = "red", lty = 2)  # Add horizontal line for exact
legend("topright", legend = c("Estimated  ", "Exact  "),
       col = c("blue", "red"), lty = c(1, 2), bty = "n")

# Plot deviations
plot(n_values, deviations, type = "l", log = "xy", col = "green",
     xlab = "Number of Samples (n)", ylab = "Deviation |  - ^|",
     main = "Deviation from Exact  ")
legend("topright", legend = c("Deviation"),
       col = c("green"), lty = c(1), bty = "n")
```



Question 2

```r
set.seed(123)  # For reproducibility

# Parameters
n <- 30  # Sample size
mu <- 10  # True mean
sigma <- 2  # True standard deviation
N_sim <- 10000  # Number of simulations
```

```r
# Initialize vectors to store the estimators
mean_estimates <- numeric(N_sim)
trimmed_mean_estimates <- numeric(N_sim)
median_estimates <- numeric(N_sim)

# Monte Carlo Simulation
for (i in 1:N_sim) {
  # Generate random sample
  sample <- rnorm(n, mean = mu, sd = sigma)

  # Compute estimators
  mean_estimates[i] <- mean(sample)  # Sample mean
  trimmed_mean_estimates[i] <- mean(sample, trim = 0.2)  # 20% trimmed mean
  median_estimates[i] <- median(sample)  # Sample median
}

# Compute biases
mean_bias <- mean(mean_estimates) - mu
trimmed_mean_bias <- mean(trimmed_mean_estimates) - mu
median_bias <- mean(median_estimates) - mu

# Display results
cat("Bias of Sample Mean: ", mean_bias, "\n")
```

```
## Bias of Sample Mean:  0.003730472
```

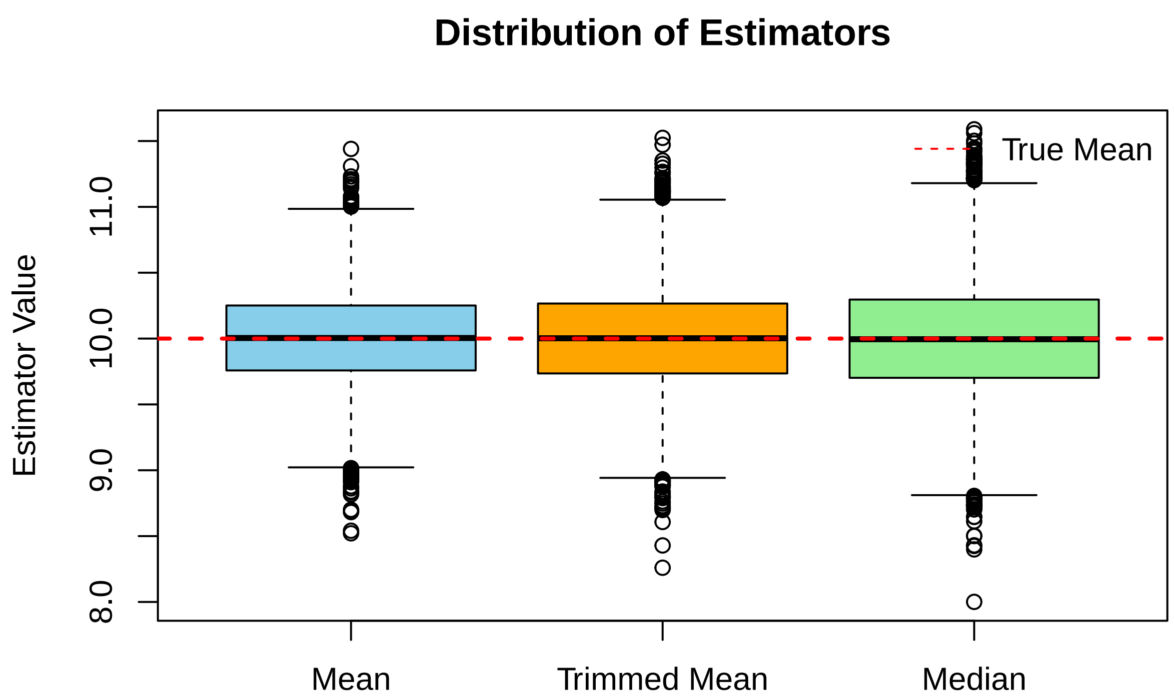
```r
cat("Bias of 20% Trimmed Mean: ", trimmed_mean_bias, "\n")
```

```
## Bias of 20% Trimmed Mean:  0.003196888
```

```r
cat("Bias of Sample Median: ", median_bias, "\n")
```

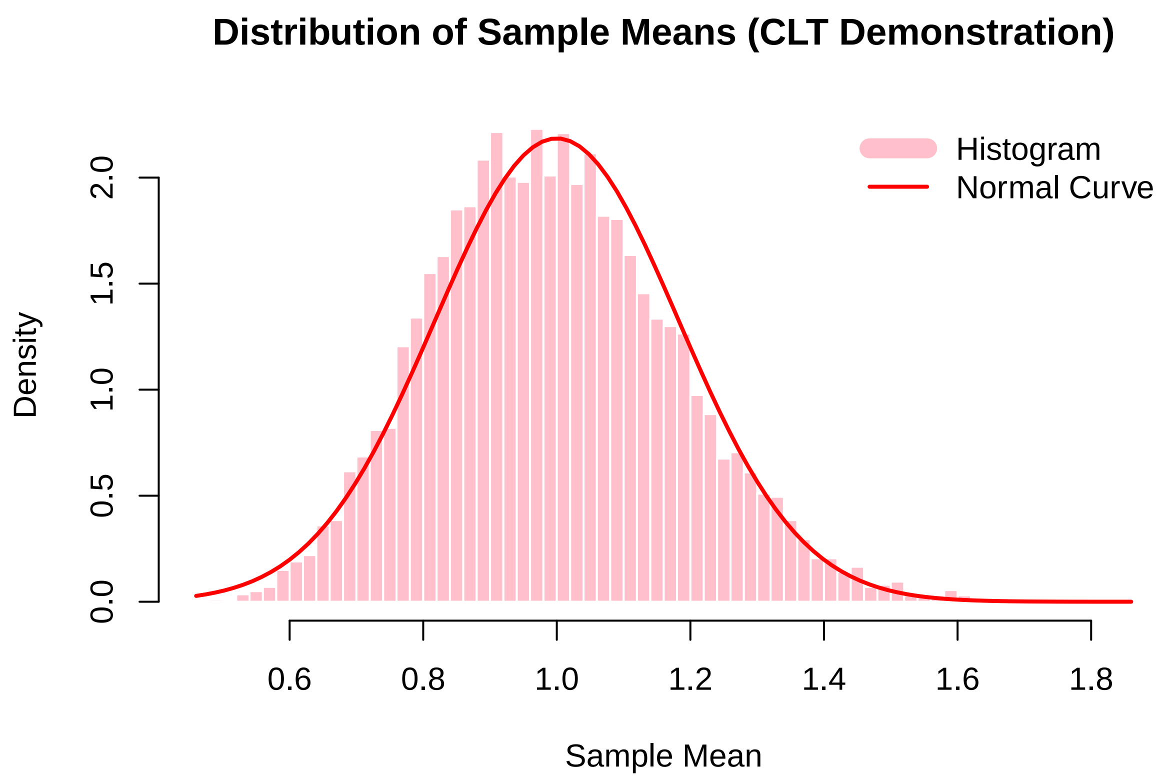
```
## Bias of Sample Median:  0.002463324
```

```r
# Visualization
boxplot(mean_estimates, trimmed_mean_estimates, median_estimates,
        names = c("Mean", "Trimmed Mean", "Median"),
        main = "Distribution of Estimators",
        ylab = "Estimator Value",
        col = c("skyblue", "orange", "lightgreen"))
abline(h = mu, col = "red", lty = 2, lwd = 2)  # True mean
legend("topright", legend = "True Mean", col = "red", lty = 2, bty = "n")
```



Question 3

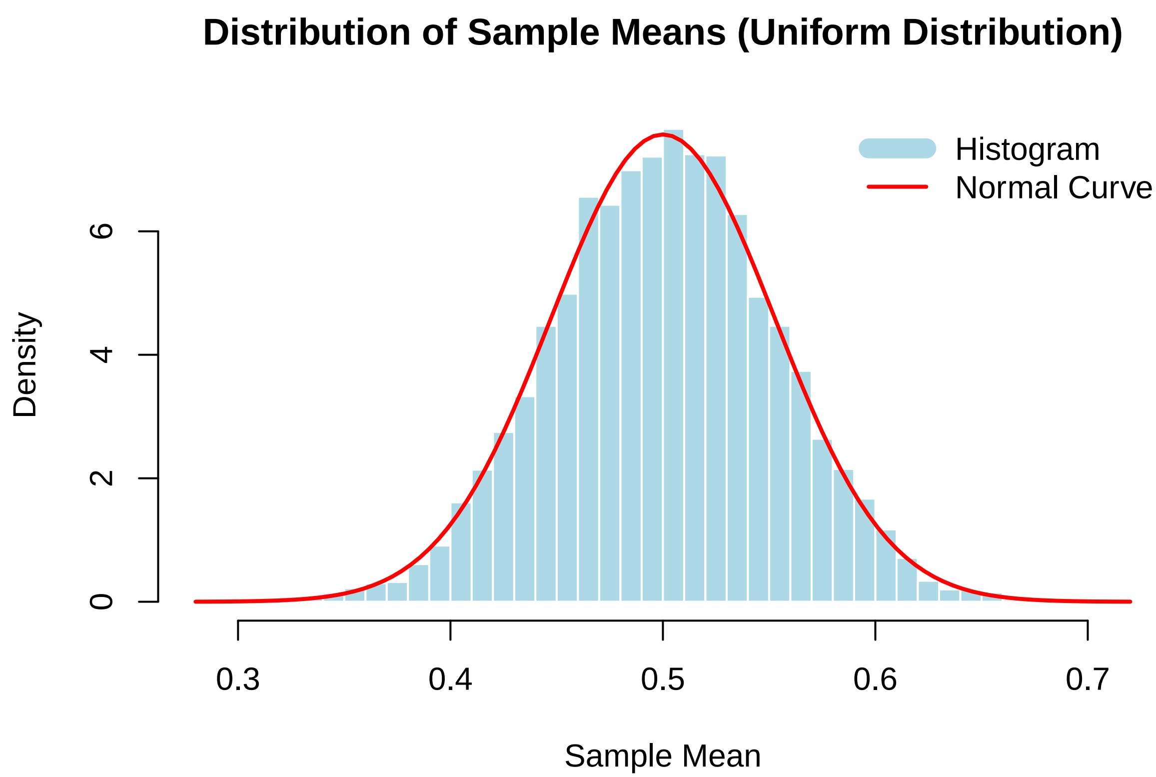
```r
set.seed(1000)  # For reproducibility

# Parameters
n <- 30  # Sample size
N_sim <- 10000  # Number of simulations

# Generate sample means from a non-normal distribution (Exponential)
sample_means <- numeric(N_sim)
for (i in 1:N_sim) {
  sample <- rexp(n, rate = 1)  # Exponential distribution with rate = 1
  sample_means[i] <- mean(sample)  # Compute sample mean
}

# Theoretical normal distribution
mean_theoretical <- 1  # Mean of Exponential(1) = 1
sd_theoretical <- 1 / sqrt(n)  # Standard deviation of the sampling distribution

# Visualization
hist(sample_means, breaks = 50, probability = TRUE,
     main = "Distribution of Sample Means (CLT Demonstration)",
     xlab = "Sample Mean", col = "pink", border = "white")
curve(dnorm(x, mean = mean_theoretical, sd = sd_theoretical),
      col = "red", lwd = 2, add = TRUE)  # Overlay normal curve
legend("topright", legend = c("Histogram", "Normal Curve"),
       col = c("pink", "red"), lty = c(1, 1), lwd = c(10, 2), bty = "n")
```



Using Exponential Distribution

```
set.seed(1000)  # For reproducibility

# Parameters
n <- 30  # Sample size
N_sim <- 10000  # Number of simulations

# Generate sample means from a uniform distribution
sample_means_uniform <- numeric(N_sim)
```

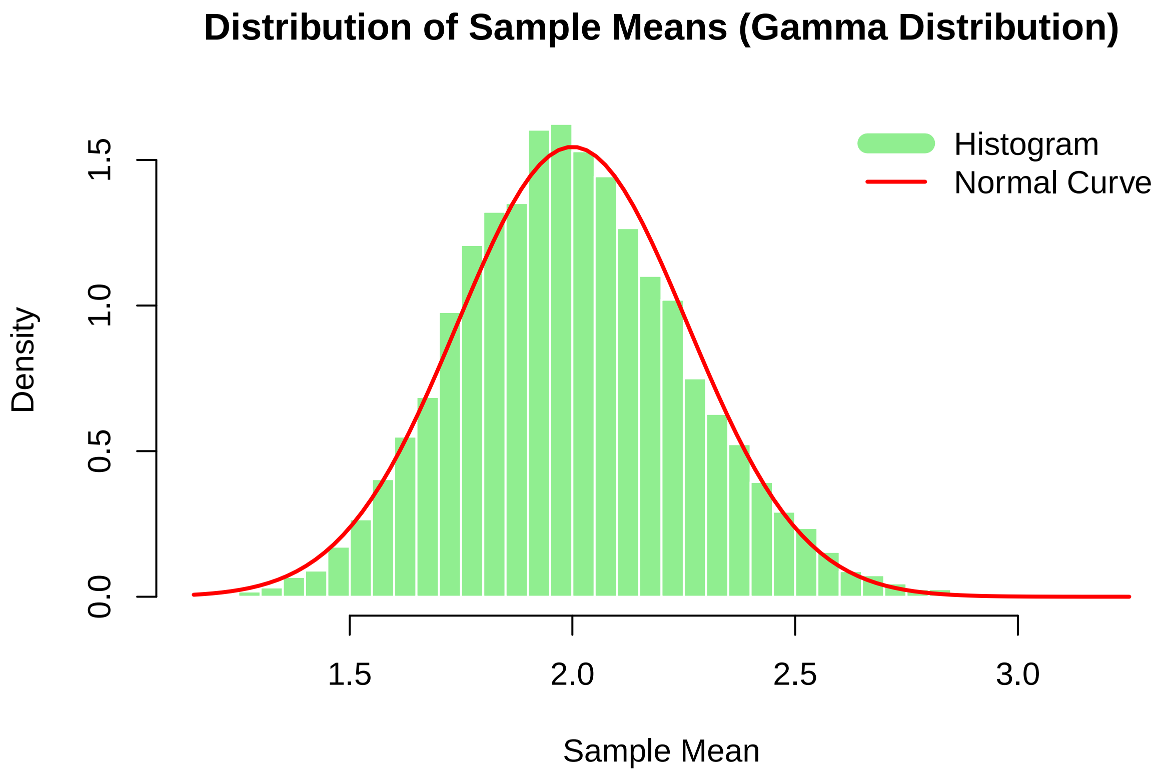
```r
for (i in 1:N_sim) {
  sample <- runif(n, min = 0, max = 1)  # Uniform distribution between 0 and 1
  sample_means_uniform[i] <- mean(sample)  # Compute sample mean
}

# Theoretical normal distribution
mean_theoretical_uniform <- 0.5  # Mean of Uniform(0, 1) = (0 + 1) / 2
sd_theoretical_uniform <- sqrt((1/12) / n)  # Standard deviation of the sampling distribution

# Visualization
hist(sample_means_uniform, breaks = 50, probability = TRUE,
     main = "Distribution of Sample Means (Uniform Distribution)",
     xlab = "Sample Mean", col = "lightblue", border = "white")
curve(dnorm(x, mean = mean_theoretical_uniform, sd = sd_theoretical_uniform),
      col = "red", lwd = 2, add = TRUE)  # Overlay normal curve
legend("topright", legend = c("Histogram", "Normal Curve"),
       col = c("lightblue", "red"), lty = c(1, 1), lwd = c(10, 2), bty = "n")
```



Using Uniform Distribution

```
set.seed(1000)  # For reproducibility

# Parameters
n <- 30  # Sample size
N_sim <- 10000  # Number of simulations

# Generate sample means from a gamma distribution
sample_means_gamma <- numeric(N_sim)
```

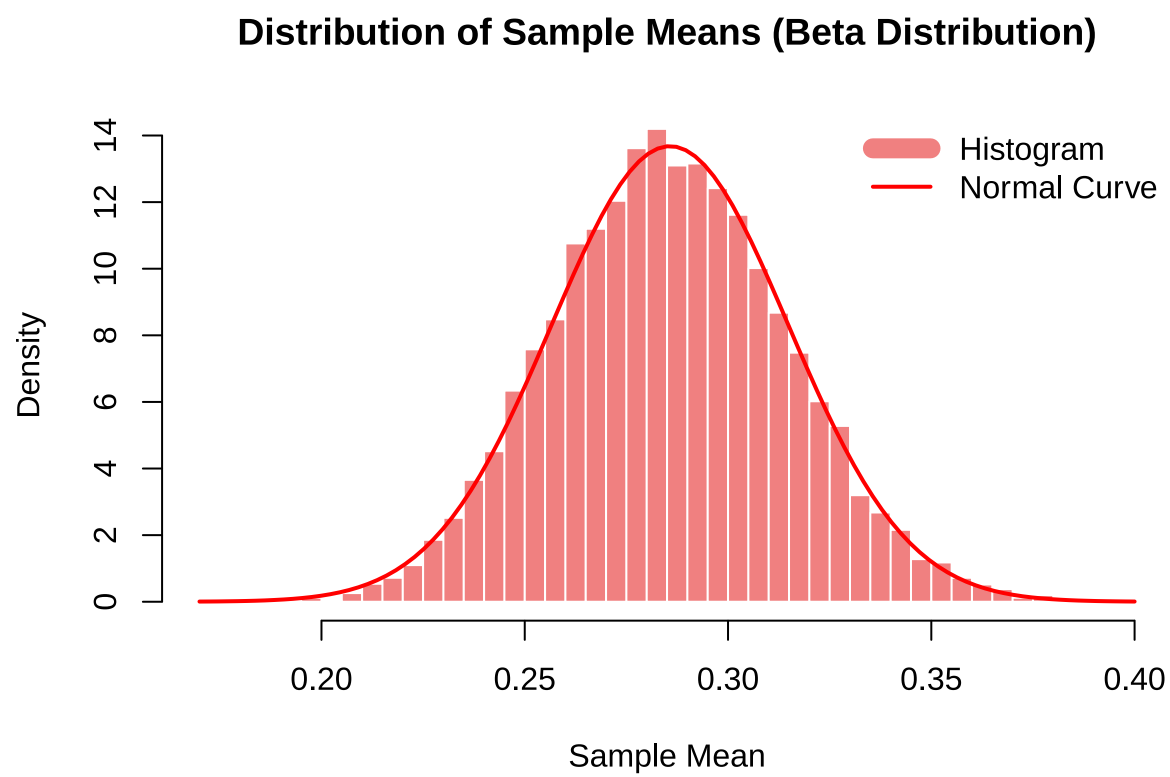
```r
for (i in 1:N_sim) {
  sample <- rgamma(n, shape = 2, rate = 1)  # Gamma distribution with shape = 2, rate = 1
  sample_means_gamma[i] <- mean(sample)  # Compute sample mean
}

# Theoretical normal distribution
mean_theoretical_gamma <- 2  # Mean of Gamma(2, 1) = shape / rate = 2
sd_theoretical_gamma <- sqrt(2 / n)  # Standard deviation of the sampling distribution

# Visualization
hist(sample_means_gamma, breaks = 50, probability = TRUE,
     main = "Distribution of Sample Means (Gamma Distribution)",
     xlab = "Sample Mean", col = "lightgreen", border = "white")
curve(dnorm(x, mean = mean_theoretical_gamma, sd = sd_theoretical_gamma),
      col = "red", lwd = 2, add = TRUE)  # Overlay normal curve
legend("topright", legend = c("Histogram", "Normal Curve"),
       col = c("lightgreen", "red"), lty = c(1, 1), lwd = c(10, 2), bty = "n")
```



Using Gamma Distribution

```
set.seed(1000)  # For reproducibility

# Parameters
n <- 30  # Sample size
N_sim <- 10000  # Number of simulations

# Generate sample means from a beta distribution
sample_means_beta <- numeric(N_sim)
```

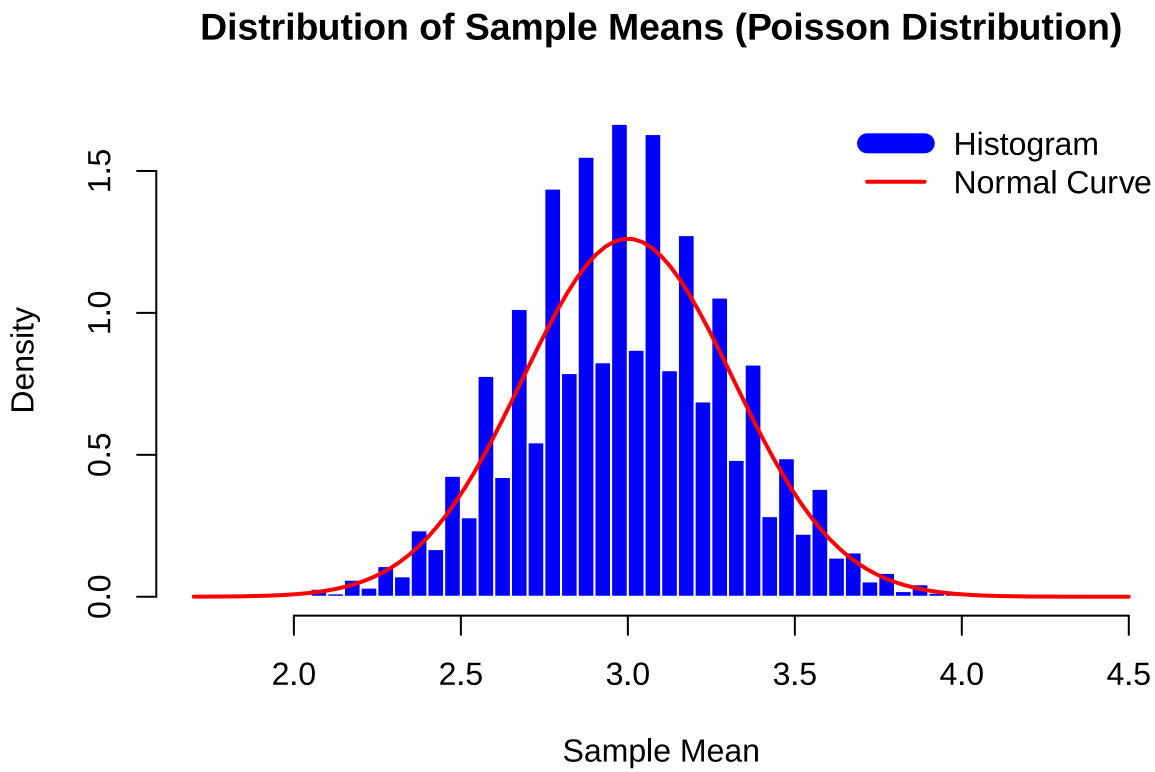
```r
for (i in 1:N_sim) {
  sample <- rbeta(n, shape1 = 2, shape2 = 5)  # Beta distribution with shape1 = 2, shape2 = 5
  sample_means_beta[i] <- mean(sample)  # Compute sample mean
}

# Theoretical normal distribution
mean_theoretical_beta <- 2 / (2 + 5)  # Mean of Beta(2, 5) = shape1 / (shape1 + shape2)
sd_theoretical_beta <- sqrt((2 * 5) / ((2 + 5)^2 * (2 + 5 + 1))) / sqrt(n)  # Standard deviation of the sampling distribution

# Visualization
hist(sample_means_beta, breaks = 50, probability = TRUE,
     main = "Distribution of Sample Means (Beta Distribution)",
     xlab = "Sample Mean", col = "lightcoral", border = "white")
curve(dnorm(x, mean = mean_theoretical_beta, sd = sd_theoretical_beta),
      col = "red", lwd = 2, add = TRUE)  # Overlay normal curve
legend("topright", legend = c("Histogram", "Normal Curve"),
       col = c("lightcoral", "red"), lty = c(1, 1), lwd = c(10, 2), bty = "n")
```



Using Beta Distribution

```
set.seed(1000)  # For reproducibility

# Parameters
n <- 30  # Sample size
N_sim <- 10000  # Number of simulations

# Generate sample means from a Poisson distribution
sample_means_poisson <- numeric(N_sim)
```

```r
for (i in 1:N_sim) {
  sample <- rpois(n, lambda = 3)  # Poisson distribution with lambda = 3
  sample_means_poisson[i] <- mean(sample)  # Compute sample mean
}

# Theoretical normal distribution
mean_theoretical_poisson <- 3  # Mean of Poisson(lambda) = lambda
sd_theoretical_poisson <- sqrt(3 / n)  # Standard deviation of the sampling distribution

# Visualization
hist(sample_means_poisson, breaks = 50, probability = TRUE,
     main = "Distribution of Sample Means (Poisson Distribution)",
     xlab = "Sample Mean", col = "blue", border = "white")
curve(dnorm(x, mean = mean_theoretical_poisson, sd = sd_theoretical_poisson),
      col = "red", lwd = 2, add = TRUE)  # Overlay normal curve
legend("topright", legend = c("Histogram", "Normal Curve"),
       col = c("blue", "red"), lty = c(1, 1), lwd = c(10, 2), bty = "n")
```



Using Poisson Distribution