Markov Chain Excercise and Solution

Md Ariful Islam

July 24, 2024

Question 1

Compute the following integral in R using Monte-Carlo method in two different ways (as you can):

$$\int_0^\infty \frac{\theta^6 \exp^{-\theta^2-\theta^3-2\theta}}{1+3\theta^2} d\theta$$

[Hint: Try to write this integral as $\int g(\theta) * h(\theta)\, d\theta$ where $g(\theta)$ is a known density that is easy to simulate from. Then, you can draw a large number of samples from $g(\theta)$, evaluate $f(\theta)$ for every one of those samples and take the mean. Here I am asking to do in two different ways so you will have to think of two diffrent choices for each of $f(\theta)$ and $g(\theta)$.

Answer:

To compute the above integral in Monte Carlo method, we need to decompose $\int g(\theta) * h(\theta)\, d\theta$ as $g(\theta)$ and $h(\theta)$ where $g(\theta)$ is a known density that is easy to simulate from.

One way is:

Let $g(\theta)$ follows exp(1), which is completely known distribution and
$h(\theta)$ becomes

$$\frac{\theta^6 \exp^{-\theta^2-\theta^3-\theta}}{1+3\theta^2}$$

```r
# Define the integral function
exact_function <- function(theta) {
  numerator <- theta^6 * exp(-theta^2 - theta^3 - 2*theta)
  denominator <- 1 + 3*theta^2
  return(numerator / denominator)
}

# Perform numerical integration
actual_result <- integrate(exact_function, lower = 0, upper = Inf)
cat("The actual value of the integral is:", actual_result$value)
```

```
## The actual value of the integral is: 0.003598568
```

```
# Integral approximation when $g(\theta)$ follows exp(1)
g_theta<-rexp(10000,1)
samples_h_theta<- g_theta^6*exp(-g_theta^2-g_theta^3-g_theta)/(1+3*g_theta^2)
mean_samples_h_theta<-mean(samples_h_theta)
cat("The  approximate value of the integral is:", mean_samples_h_theta)
```

```
## The  approximate value of the integral is: 0.003625605
```

Onether way is:

Let $g(\theta)$ follows gamma(7,2), which is completely known distribution and
$h(\theta)$ becomes

$$\frac{\theta^6 \exp^{-\theta^2-\theta^3}}{2(1+3\theta^2)}$$

```
# Integral approximation when $g(\theta)$ follows gamma(1,2)
g_theta<-rgamma(10000,shape=7,rate=2)
samples_h_theta<-(gamma(7)/128)*exp(-g_theta^2-g_theta^3)/((1+3*g_theta^2))
mean_samples_h_theta<-mean(samples_h_theta)
cat("The  approximate value of the integral is:", mean_samples_h_theta)
```

```
## The  approximate value of the integral is: 0.0034095
```

Question 2

Evaluate the following integral in R using Monte-Carlo Method:

$$\int_{1.5}^{6} \frac{\log\theta}{2+\theta^2} d\theta$$

Answer:

Let $g(\theta)$ follows unif(1.5, 6), which is completely known distribution and
$h(\theta)$ becomes

$$\frac{4.5\log\theta}{2+\theta^2}$$

```
# Define the integral function
exact_function <- function(theta) {
  numerator <- log(theta)
  denominator <- 2 + theta^2
  return(numerator / denominator)
}

# Perform numerical integration
actual_result <- integrate(exact_function, lower = 1.5, upper = 6)
cat("The actual value of the integral is:", actual_result$value)
```

```
## The actual value of the integral is: 0.3734026
```

```r
# Integral approximation when $g(\theta)$ follows exp(1)
g_theta<-runif(10000,1.5, 6)
samples_h_theta<- (4.5*log(g_theta))/(2+g_theta^2)
mean_samples_h_theta<-mean(samples_h_theta)
cat("The  approximate value of the integral is:", mean_samples_h_theta)
```

```
## The  approximate value of the integral is: 0.375375
```

Question 3

Our target distribution is $\theta \sim$ Gamma(2,4) distribution. Run M-H algorithm to simulate from this target distribution in two different ways with the following two choices of proposal distribution. [Of course, we know how to draw from Gamma(2,4) directly, so no need to use M- H, but we want to do it using M-H as a proof-of-concept just to convince ourselves that M-H works.] In each case, run 50000 steps in R. report the acceptance rate (how many times out of these 50000 steps, you could accept the proposal). Start with any initial value of your choice.

(i) Normal distribution Proposal with variance = 0.1 and mean at previous value of θ
(ii) Normal distribution proposal with variance = 2.0 and mean at previous value of θ

Answer:

Here, target distribution $\theta \sim$ Gamma(2,4). We need to simulate θ from this target distribution in two ways with the specified two choices of proposal distribution.

```r
set.seed(1234)
alpha <- 2
beta <- 4

# Define the target density function
target_dist <- function(x) {
  dgamma(x, shape = alpha, rate = beta)
}

# True mean of target distribution
true_mean <- alpha / beta
cat("True mean of target distribution:", true_mean, "\n")
```

```
## True mean of target distribution: 0.5
```

```r
# Metropolis-Hastings MCMC algorithm
mcmc <- function(init_theta, iterations, proposal_sd) {
  theta <- numeric(iterations)
  theta[1] <- init_theta
  accepted_proposal <- 0

  for (i in 2:iterations) {
    generate_theta <- rnorm(1, mean = theta[i - 1], sd = proposal_sd)
    proposed_target_dist <- target_dist(generate_theta)
    current_target_dist <- target_dist(theta[i - 1])
    acceptance_ratio <- proposed_target_dist / current_target_dist

    if (runif(1) < acceptance_ratio) {
      theta[i] <- generate_theta
      accepted_proposal <- accepted_proposal + 1
    } else {
      theta[i] <- theta[i - 1]
    }
  }

  return(list(theta = theta, accepted_proposal = accepted_proposal))
}

#  using proposal variance = 0.1
# Set parameters
init_theta <- 1
iterations <- 50000
proposed_std <- sqrt(0.1)

# Run the MCMC algorithm
results <- mcmc(init_theta, iterations, proposed_std)
all_gen_theta <- results$theta
approx_mean<-mean(all_gen_theta)
acceptance_rate <- results$accepted_proposal / (iterations)

cat("Approximate mean:", approx_mean, "\n")
```

```
## Approximate mean: 0.5021729
```

```r
cat("Acceptance rate:", acceptance_rate, "\n")
```

```
## Acceptance rate: 0.6671
```

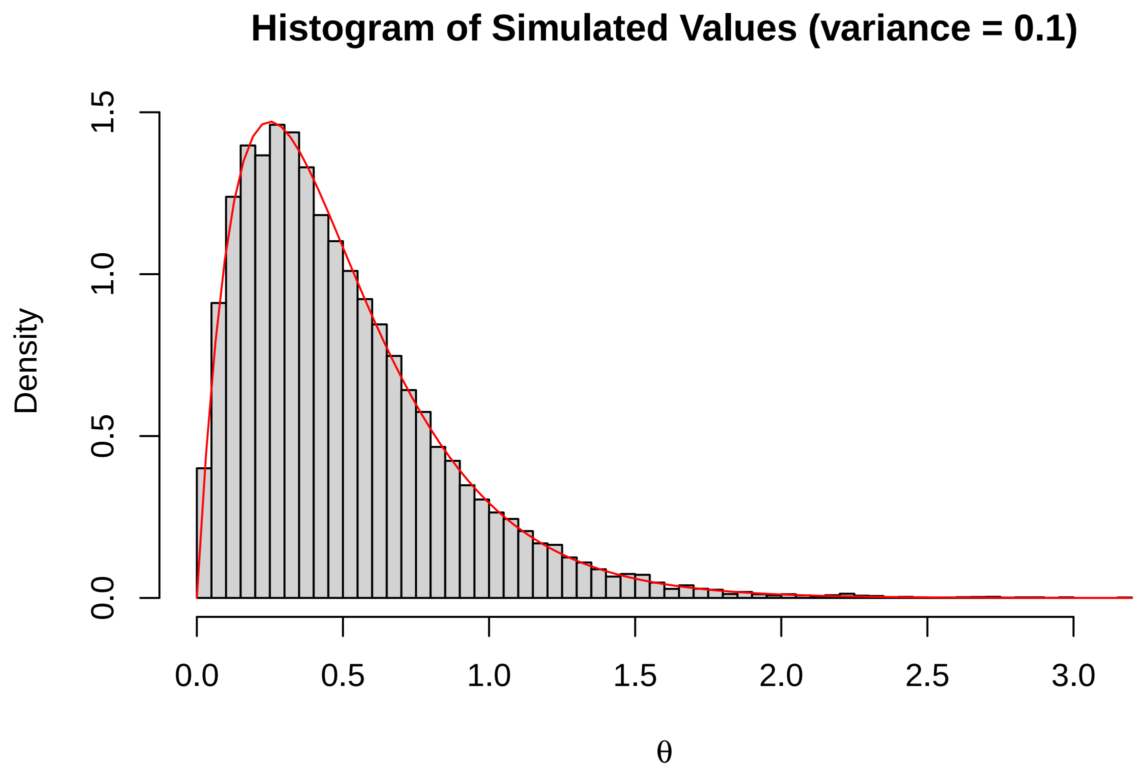
```r
burn_in <- 10000
theta_burned <- results$theta[(burn_in + 1):iterations]
head(theta_burned)
```

```
## [1] 0.1944858 0.6883976 0.3912349 0.3912349 0.5281308 0.5906119
```

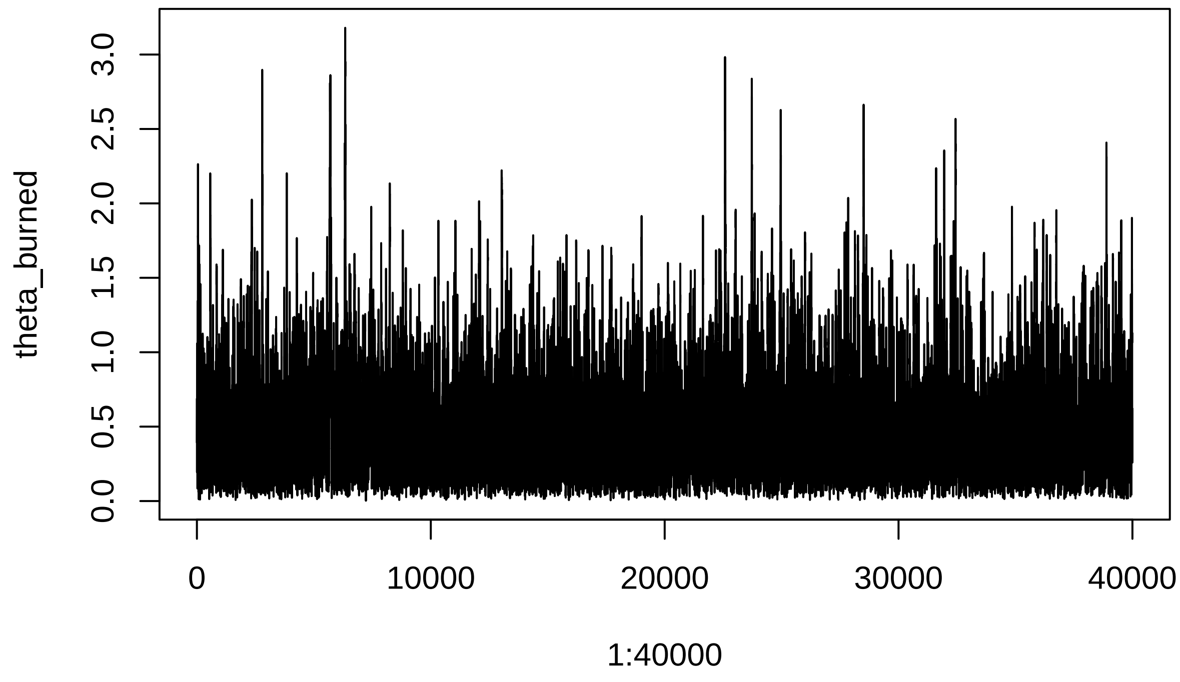
```
estimated_mean <- mean(theta_burned)
cat("Estimated mean (variance = 0.1):", estimated_mean, "\n")
```

```
## Estimated mean (variance = 0.1): 0.5016757
```

```
# Plot the histogram of the remaining sampled values
hist(theta_burned, breaks = 50, probability = TRUE
     , main = "Histogram of Simulated Values (variance = 0.1)"
     , xlab = expression(theta))
curve(dgamma(x, shape = 2, rate = 4), add = TRUE, col = "red")
```



```r
plot(1:40000,theta_burned, "l")
```



```r
# using proposal variance = 2
# Set parameters
init_theta <- 1
iterations <- 50000
proposed_std <- sqrt(2)

# Run the MCMC algorithm
results <- mcmc(init_theta, iterations, proposed_std)
all_gen_theta <- results$theta
approx_mean<-mean(all_gen_theta)
acceptance_rate <- results$accepted_proposal / (iterations)

cat("Approximate mean:", approx_mean, "\n")
```

```
## Approximate mean: 0.5034745
```

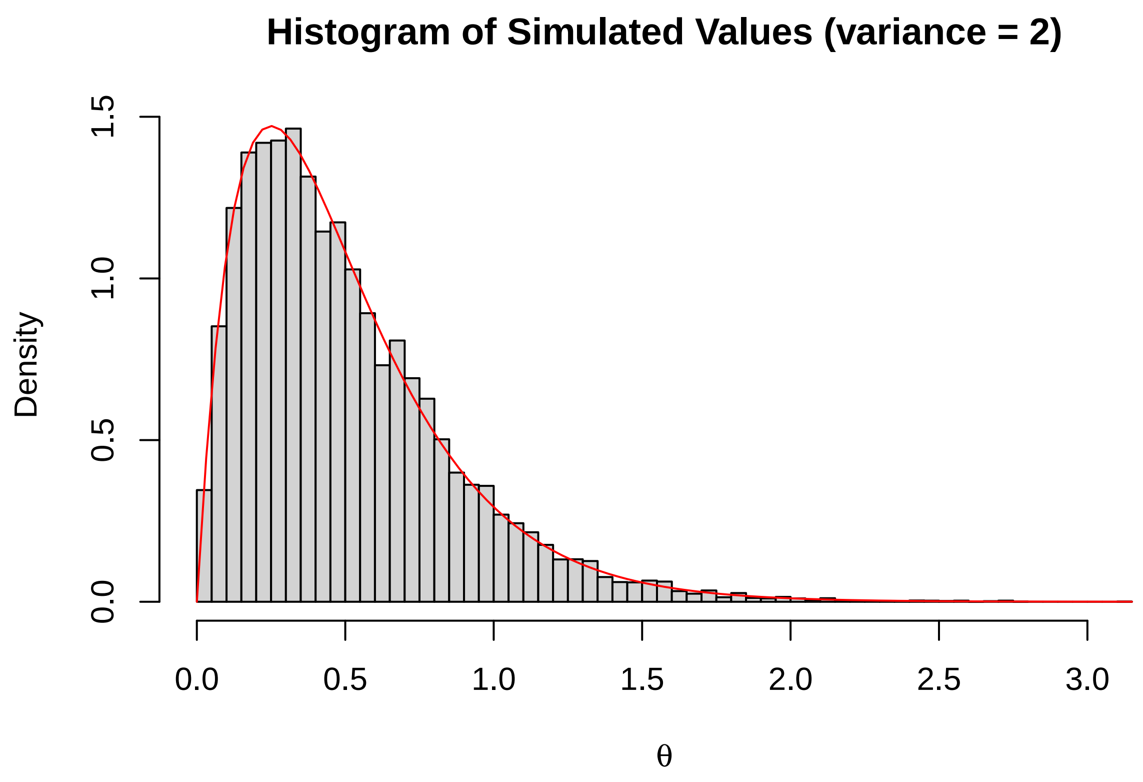
```r
cat("Acceptance rate:", acceptance_rate, "\n")
```

```
## Acceptance rate: 0.2453
```

```r
burn_in <- 10000
theta_burned <- results$theta[(burn_in + 1):iterations]
estimated_mean <- mean(theta_burned)
cat("Estimated mean (variance = 2):", estimated_mean, "\n")
```

```
## Estimated mean (variance = 2): 0.5042267
```

```r
# Plot the histogram of the remaining sampled values
hist(theta_burned, breaks = 50, probability = TRUE
     , main = "Histogram of Simulated Values (variance = 2)"
     , xlab = expression(theta))
curve(dgamma(x, shape = 2, rate = 4), add = TRUE, col = "red")
```



Question 4

Why the two proposals mentioned in Q3 are technically valid but practically not very reasonable to use as M-H proposal when the target distribution is Gamma(2,4)? [Think conceptually]

Answer:

Practically, a proposal with a small variance (0.1) may result in high acceptance rates and dense mixing due to small step sizes, making it inefficient.

A proposal with a large variance (2.0) may result in a high rejection rate because proposals are often far from the current value, making it inefficient as well.

Question 5

Repeat Q3 with the same target distribution but with a new proposal chosen from a lognormal distribution, so that, at step (t + 1),

$$\log \theta^* \sim N(\log \theta^{(t)}, \tau^2)$$

First, calculate in pen and paper the formula of acceptance probability. Then, in R, draw 50000 observations using M-H algorithm with above proposal with $\tau^2 = 0.25$. Report the acceptance rate. Then, repeat with $\tau^2 = 25$. Report the acceptance rate.

Answer:

```r
set.seed(1234)
alpha <- 2
beta <- 4


# True mean of target distribution
true_mean <- alpha / beta
cat("True mean of target distribution:", true_mean, "\n")
```

```
## True mean of target distribution: 0.5
```

```r
# Metropolis-Hastings MCMC algorithm
mcmc <- function(init_theta, iterations, proposal_sd) {
  theta <- numeric(iterations)
  theta[1] <- init_theta
  accepted_proposal <- 0

  for (i in 2:iterations) {
    generate_theta <- rlnorm(1, mean = log(theta[i - 1]), sd = proposal_sd)
    acceptance_ratio<-(dgamma(generate_theta, shape = 2, rate = 4)
                       * dlnorm(theta[i-1], log(generate_theta)
                                , sqrt(proposed_std))) /
      (dgamma(theta[i-1], shape = 2, rate = 4)
       * dlnorm(generate_theta, log(theta[i-1]), sqrt(proposed_std)))

    if (runif(1) < acceptance_ratio) {
      theta[i] <- generate_theta
      accepted_proposal <- accepted_proposal + 1
    } else {
      theta[i] <- theta[i - 1]
    }
  }

  return(list(theta = theta, accepted_proposal = accepted_proposal))
}

#  using proposal variance = 0.25
# Set parameters
init_theta <- 1
iterations <- 50000
proposed_std <- sqrt(.25)

# Run the MCMC algorithm
results <- mcmc(init_theta, iterations, proposed_std)
all_gen_theta <- results$theta
approx_mean<-mean(all_gen_theta)
acceptance_rate <- results$accepted_proposal / (iterations)

cat("Approximate mean:", approx_mean, "\n")
```

```
## Approximate mean: 0.4976357
```

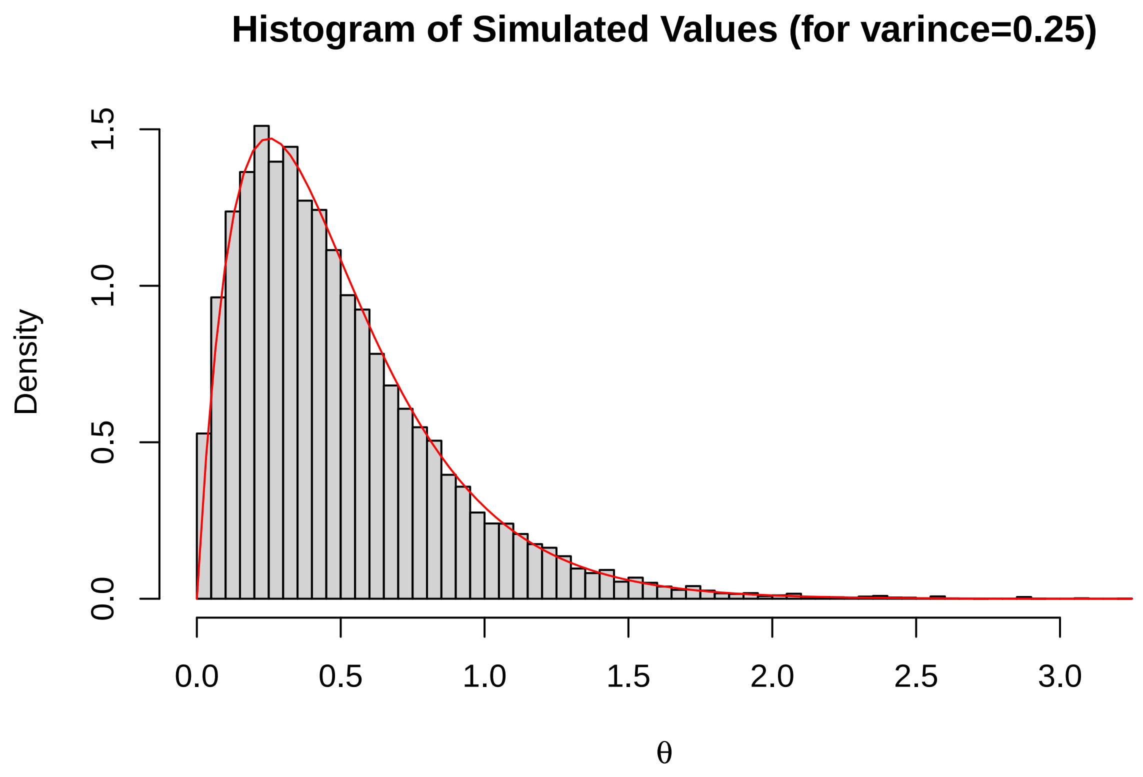
```
cat("Acceptance rate:", acceptance_rate, "\n")
```

```
## Acceptance rate: 0.79064
```

```
# Plot the histogram of the sampled values
burn_in <- 10000
theta_burned <- all_gen_theta[(burn_in + 1):iterations]
estimated_mean <- mean(theta_burned)
cat("Estimated mean (for varince=0.25):", estimated_mean, "\n")
```

```
## Estimated mean (for varince=0.25): 0.4953533
```

```
hist(theta_burned, breaks = 50, probability = TRUE
     , main = "Histogram of Simulated Values (for varince=0.25)"
     , xlab = expression(theta))
curve(dgamma(x, shape = 2, rate = 4), add = TRUE, col = "red")
```



```r
# using proposal variance = 25
# Set parameters
init_theta <- 1
iterations <- 50000
proposed_std <- sqrt(25)

# Run the MCMC algorithm
results <- mcmc(init_theta, iterations, proposed_std)
all_gen_theta <- results$theta
approx_mean<-mean(all_gen_theta)
acceptance_rate <- results$accepted_proposal / (iterations)

cat("Approximate mean:", approx_mean, "\n")
```

```
## Approximate mean: 0.5006482
```

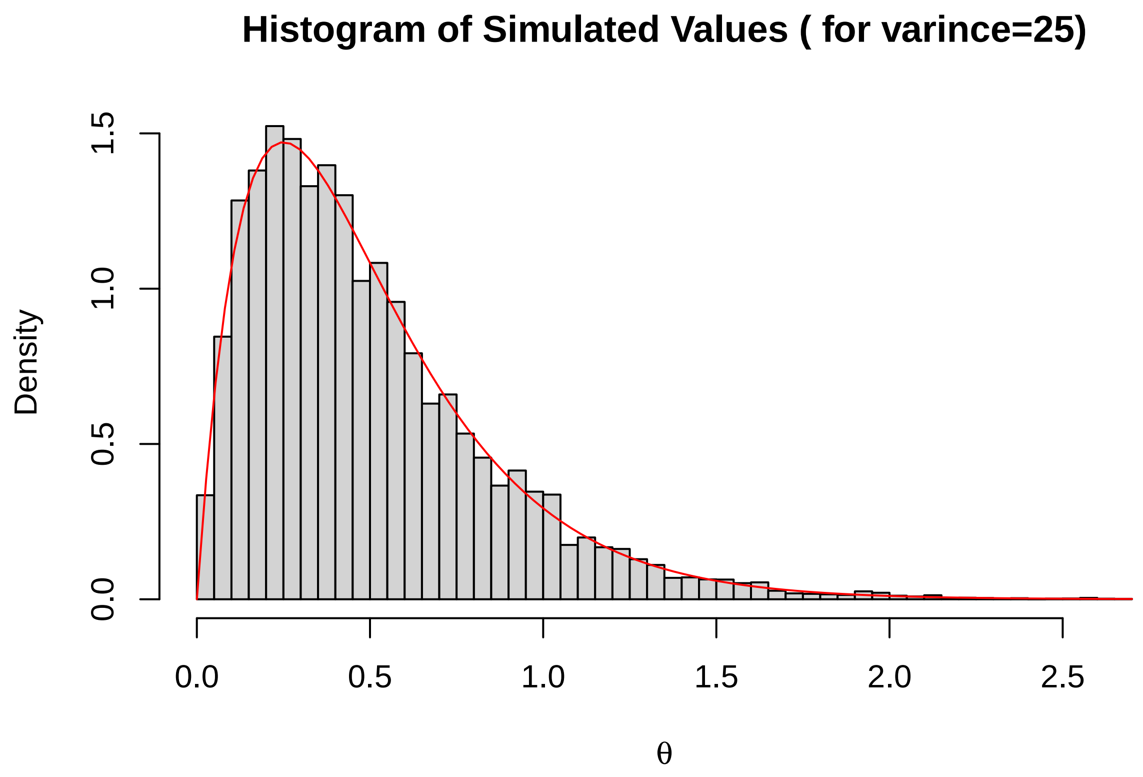
```r
cat("Acceptance rate:", acceptance_rate, "\n")
```

```
## Acceptance rate: 0.1878
```

```r
# Plot the histogram of the sampled values
burn_in <- 10000
theta_burned <- all_gen_theta[(burn_in + 1):iterations]
estimated_mean <- mean(theta_burned)
cat("Estimated mean (for varince=25):", estimated_mean, "\n")
```

```
## Estimated mean (for varince=25): 0.4995207
```

```r
hist(theta_burned, breaks = 50, probability = TRUE
     , main = "Histogram of Simulated Values ( for varince=25)"
     , xlab = expression(theta))
curve(dgamma(x, shape = 2, rate = 4), add = TRUE, col = "red")
```



Question 6

We want to generate observations from a parametric model given by:

$$f(\theta) = c * \theta^3 \quad \text{for } \theta \in (0, 1)$$

(i) Determine c.
(ii) Write the R code to generate 10000 observations using inverse CDF method.

(iii) Write the R code to generate 10000 observations using rejection sampling
(iv) Report the average number of trials per accepted observation in Part (iii).
(v) Compare the two sets of observations generated in Part (ii)
and Part (iii) in terms of empirical mean, variance and 60% quantile.

(i) Determine c.

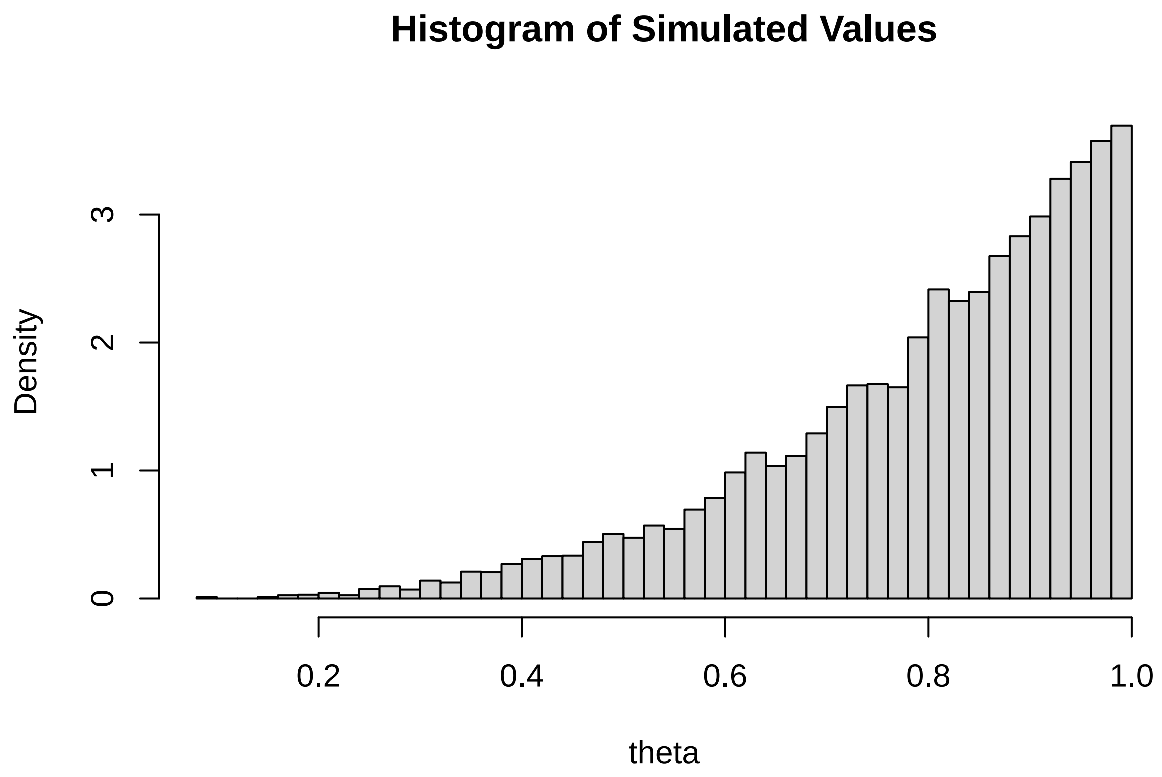
```r
# Define the integral function
exact_function <- function(theta) {
  theta^3
}

# Perform numerical integration
result <- integrate(exact_function, lower = 0, upper = 1)
integral_value=result$value
c_result=1/integral_value
cat("The value of the c is:", c_result)
```

```
## The value of the c is: 4
```

(ii) Write the R code to generate 10000 observations using inverse CDF method.

```r
set.seed(123)
# Number of observations
n <- 10000
# Generate uniform random numbers
u <- runif(n)
# Apply the inverse CDF to generate observations
observations_inverse_cdf <- u^(1/4)
hist(observations_inverse_cdf, breaks = 50, probability = TRUE
     , main = "Histogram of Simulated Values"
     , xlab = "theta")
```



(iii) Write the R code to generate 10000 observations using rejection sampling, and
(iv) Report the average number of trials per accepted observation in Part (iii).

```
# Set the seed for reproducibility
set.seed(123)

# Number of observations required
n <- 10000

# Initialize vector for accepted samples
observations_rejection <- numeric(n)

# Counter for the number of accepted samples
accepted <- 0

# Counter for the number of trials
trials <- 0

while (accepted < n) {
  # Generate candidate from proposal distribution (uniform in (0, 1))
  theta_candidate <- runif(1)
  # Generate uniform random number for acceptance-rejection criterion
  u <- runif(1)

  # Evaluate target density
  f_theta <- 4 * theta_candidate^3

  # Evaluate proposal density (uniform, which is 1 in (0, 1))
  g_theta <- 1

  # Acceptance-rejection criterion
  if (u < f_theta / (4 * g_theta)) {
    observations_rejection[accepted + 1] <- theta_candidate
    accepted <- accepted + 1
  }

  trials <- trials + 1
}

# Compute the average number of trials per accepted observation
average_trials <- trials / n
cat("average_trials:", average_trials, "\n")
```

```
## average_trials: 3.9879
```

```
# Display the first few observations
cat("Five values of observations_rejection:", head(observations_rejection), "\n")
```

```
## Five values of observations_rejection: 0.9404673 0.9568333 0.8895393 0.7584595 0.7989248 0.8
```

(v) Compare the two sets of observations generated in Part (ii) and Part (iii) in terms of empirical mean, variance and 60% quantile.

```
# Empirical mean, variance, and 60% quantile for inverse CDF method
mean_inverse_cdf <- mean(observations_inverse_cdf)
variance_inverse_cdf <- var(observations_inverse_cdf)
quantile_inverse_cdf <- quantile(observations_inverse_cdf, 0.60)

# Empirical mean, variance, and 60% quantile for rejection sampling
mean_rejection <- mean(observations_rejection)
variance_rejection <- var(observations_rejection)
quantile_rejection <- quantile(observations_rejection, 0.60)

# Print results
cat("Inverse CDF Method:\n")
```

```
## Inverse CDF Method:
```

```r
cat("Mean:", mean_inverse_cdf, "\n")
```

```
## Mean: 0.7991353
```

```r
cat("Variance:", variance_inverse_cdf, "\n")
```

```
## Variance: 0.02655274
```

```r
cat("60% Quantile:", quantile_inverse_cdf, "\n\n")
```

```
## 60% Quantile: 0.8783691
```

```r
cat("Rejection Sampling Method:\n")
```

```
## Rejection Sampling Method:
```

```r
cat("Mean:", mean_rejection, "\n")
```

```
## Mean: 0.8007772
```

```r
cat("Variance:", variance_rejection, "\n")
```

```
## Variance: 0.02589696
```

```r
cat("60% Quantile:", quantile_rejection, "\n")
```

```
## 60% Quantile: 0.8782644
```