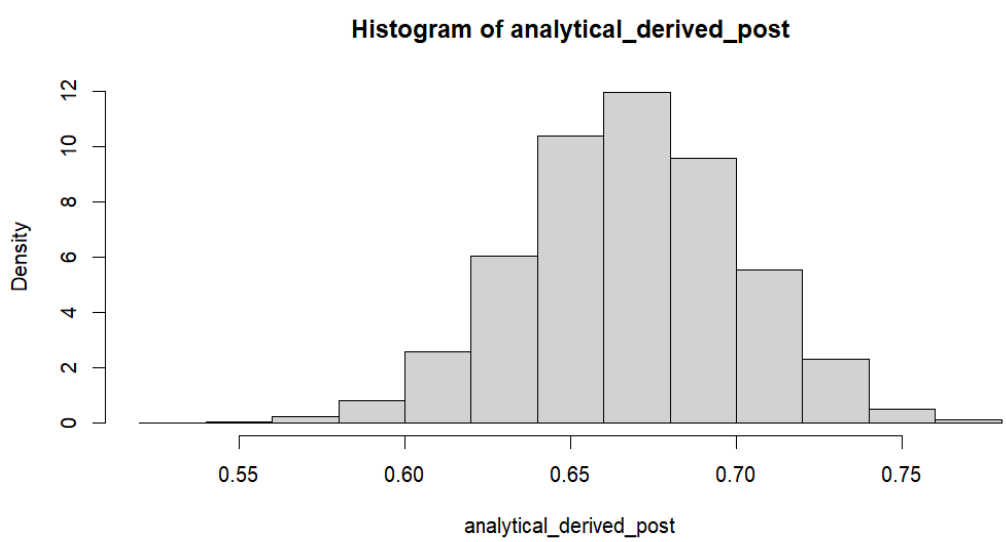
CGS-698C: Assignment 3
Submitted_by: Anil Yadav(210138)

****

PART-1, Estimating the posterior distribution using different computational methods

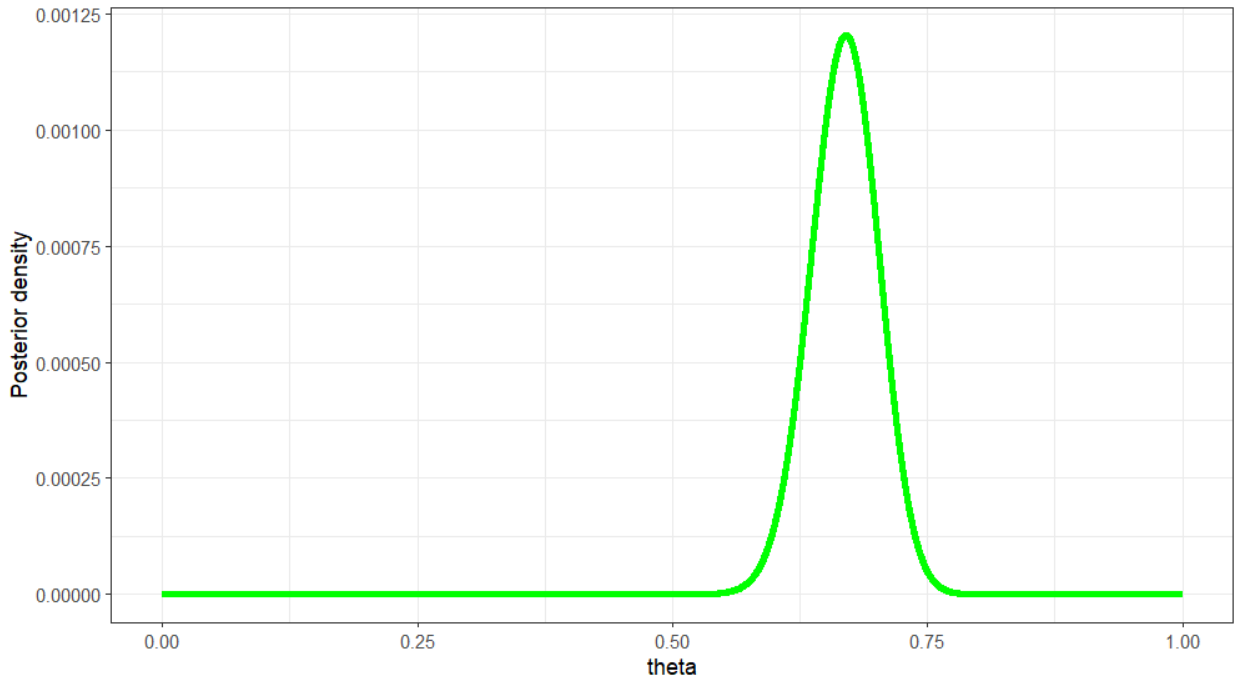
1.1

```
analytical_derived_post <- rbeta(10000,135,67)
hist(analytical_derived_post,freq = FALSE)
```



1.2

```
library(dplyr)
library(ggplot2)
y <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)

theta_grid <- seq(0,1,length=10000)

df.posterior <- data.frame(matrix(ncol=3,nrow=length(theta_grid)))
colnames(df.posterior) <- c("theta","likelihood","prior")
for(i in 1:length(theta_grid)){
  likelihood <- prod(dbinom(y,size=20,theta_grid[i]))
  prior <- dbeta(theta_grid[i],1,1)
  df.posterior[i,] <- c(theta_grid[i],likelihood,prior)
}
#Approximate marginal likelihood

df.posterior$ML <- rep(sum(df.posterior$likelihood*df.posterior$prior),10000)
#Estimate posterior density at each grid point
df.posterior <- df.posterior %>%
  mutate(posterior=likelihood*prior/ML)

ggplot(df.posterior,aes(x=theta,y=posterior))+
  geom_line(aes(group=NA),size=1.5,colour="green")+
  theme_bw()+
  scale_x_continuous()+
  ylab("Posterior density")
```



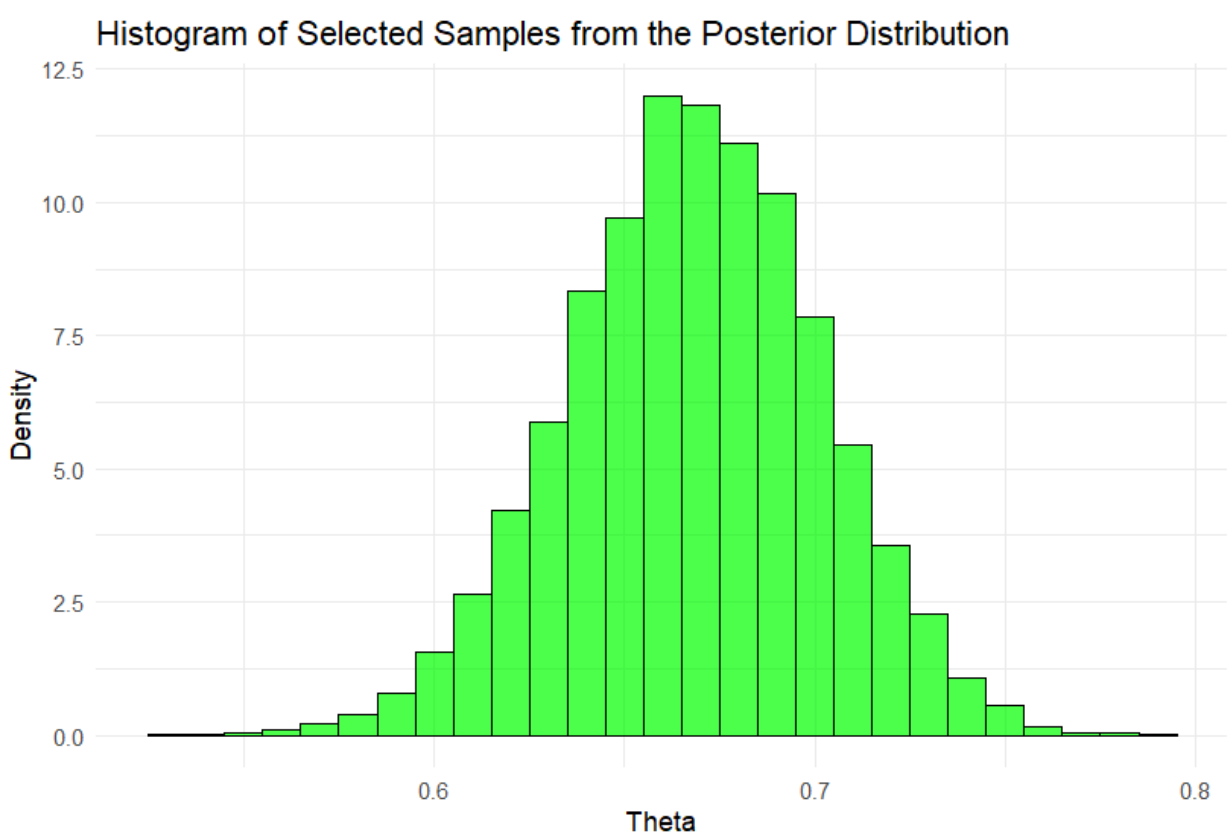
1.3

```r
library(dplyr)
library(ggplot2)

# Given data points
data <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
n <- 20  # number of trials

# Number of samples for Monte Carlo integration
num_samples <- 100000

# Draw samples from the prior Beta(1, 1)
theta_samples <- rbeta(num_samples, 1, 1)

# Compute the likelihood for each sample
likelihoods <- sapply(theta_samples, function(theta) {
  prod(dbinom(data, n, theta))
})

# Estimate the marginal likelihood by averaging the likelihoods
marginal_likelihood <- mean(likelihoods)

# Print the estimated marginal likelihood
print(paste("Estimated Marginal Likelihood:", marginal_likelihood))

```

[1] "Estimated Marginal Likelihood: 1.40880924928048e-10"

1.4

```r
library(dplyr)
library(ggplot2)

# Given data points
data <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
n <- 20  # number of trials

# Number of samples for importance sampling
num_samples <- 100000

# Proposal distribution q(theta): Beta(2, 2)
proposal_samples <- rbeta(num_samples, 2, 2)

# Compute the likelihood for each sample
likelihoods <- sapply(proposal_samples, function(theta) {
  prod(dbinom(data, n, theta))
})

# Compute the prior density for each sample: Beta(1, 1)
prior_densities <- dbeta(proposal_samples, 1, 1)

# Compute the proposal density for each sample: Beta(2, 2)
proposal_densities <- dbeta(proposal_samples, 2, 2)

# Compute weights
weights <- likelihoods * prior_densities / proposal_densities

# Normalize weights
weights <- weights / sum(weights)

# Create a dataframe to store samples and weights
df <- data.frame(theta = proposal_samples, weights = weights)

# Select N/4 samples based on their weights
selected_samples <- sample(df$theta, size = num_samples / 4, prob = df$weights, replace = TRUE)

# Print some of the selected samples
print(head(selected_samples))


# Plot histogram of the selected samples to visualize the posterior
ggplot(data.frame(theta = selected_samples), aes(x = theta)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.01, fill = "green", color = "black", alpha = 0.7) +
  labs(title = "Histogram of Selected Samples from the Posterior Distribution", x = "Theta", y = "Density") +
  theme_minimal()
```

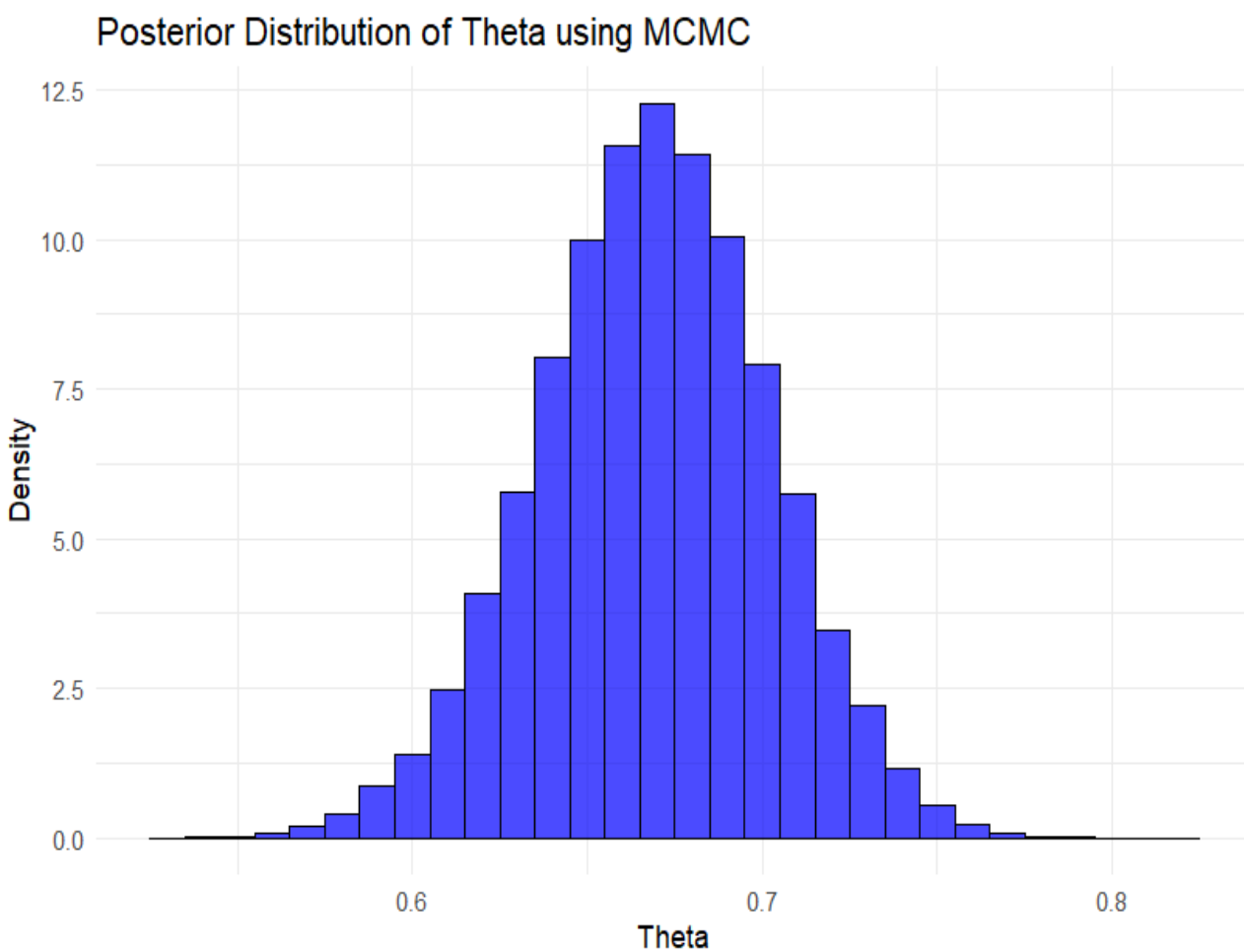


1.5

RStudio Source Editor

Untitled1*

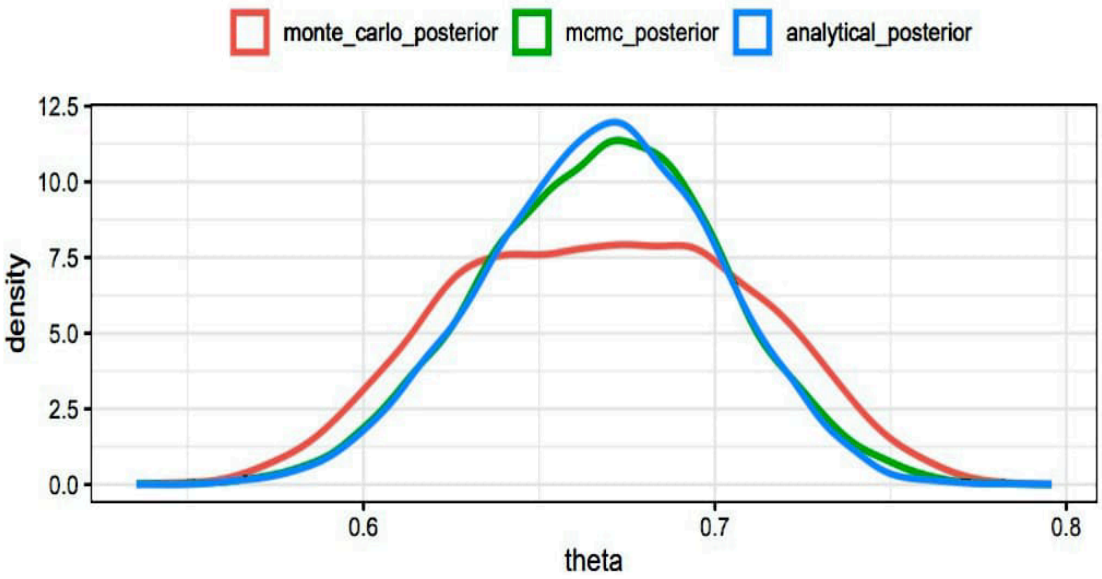
```r
# Store samples
theta_samples <- numeric(num_iterations)

# Likelihood function
likelihood <- function(theta) {
  prod(dbinom(data, n, theta))
}

# Prior density function (Beta(1, 1))
prior <- function(theta) {
  dbeta(theta, 1, 1)
}

# Proposal function (normal distribution centered around the current theta)
proposal_sd <- 0.05

for (i in 1:num_iterations) {
  # Propose a new value
  theta_proposed <- rnorm(1, mean = theta_current, sd = proposal_sd)

  # Ensure the proposed value is within the valid range [0, 1]
  if (theta_proposed < 0 || theta_proposed > 1) {
    theta_samples[i] <- theta_current
    next
  }

  # Compute acceptance probability
  acceptance_ratio <- (likelihood(theta_proposed) * prior(theta_proposed)) /
    (likelihood(theta_current) * prior(theta_current))
  alpha <- min(1, acceptance_ratio)

  # Accept or reject the proposed value
  if (runif(1) < alpha) {
    theta_current <- theta_proposed
  }

  # Store the current value
  theta_samples[i] <- theta_current
}

# Burn-in period
burn_in <- 10000
theta_samples <- theta_samples[(burn_in + 1):num_iterations]

# Plot histogram of the samples to visualize the posterior distribution
ggplot(data.frame(theta = theta_samples), aes(x = theta)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.01, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Posterior Distribution of Theta using MCMC", x = "Theta", y = "Density") +
  theme_minimal()
```



1.6

```r
library(ggplot2)

# Given data points
data <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
n <- 20  # number of trials

# Number of iterations for MCMC
num_iterations <- 100000

# Initialize theta
theta_current <- 0.5

# Store samples
theta_samples <- numeric(num_iterations)

# Likelihood function
likelihood <- function(theta) {
  prod(dbinom(data, n, theta))
}

# Prior density function (Beta(1, 1))
prior <- function(theta) {
  dbeta(theta, 1, 1)
}

# Proposal function (normal distribution centered around the current theta)
proposal_sd <- 0.05

for (i in 1:num_iterations) {
  # Propose a new value
  theta_proposed <- rnorm(1, mean = theta_current, sd = proposal_sd)

  # Ensure the proposed value is within the valid range [0, 1]
  if (theta_proposed < 0 || theta_proposed > 1) {
    theta_samples[i] <- theta_current
    next
  }

  # Compute acceptance probability
  acceptance_ratio <- (likelihood(theta_proposed) * prior(theta_proposed)) /
    (likelihood(theta_current) * prior(theta_current))
  alpha <- min(1, acceptance_ratio)

  # Accept or reject the proposed value
  if (runif(1) < alpha) {
    theta_current <- theta_proposed
  }

  # Store the current value
  theta_samples[i] <- theta_current
}

# Burn-in period
burn_in <- 10000
theta_samples <- theta_samples[(burn_in + 1):num_iterations]

# Plot histogram of the samples to visualize the posterior distribution
ggplot(data.frame(theta = theta_samples), aes(x = theta)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.01, fill = "blue", color = "black", alpha = 0.7)
  labs(title = "Posterior Distribution of Theta using MCMC", x = "Theta", y = "Density") +
  theme_minimal()
```

38:9 (Top Level)



****

Part 3: Hamiltonian Monte Carlo sampler

****

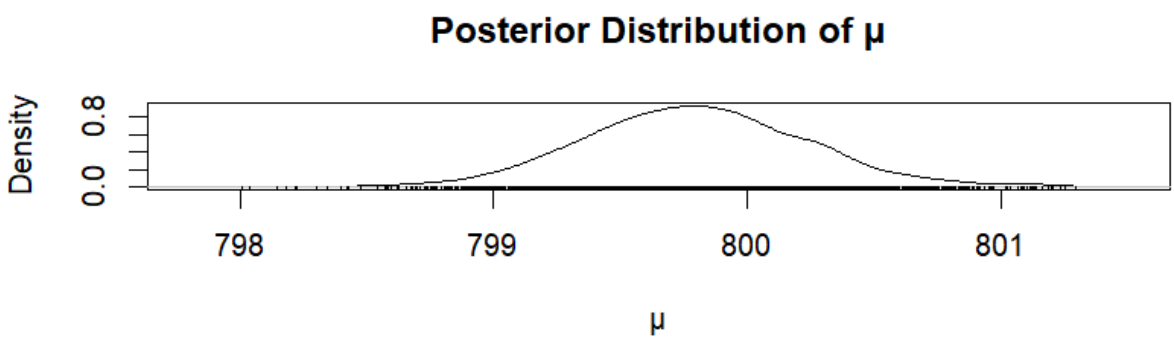
3.1

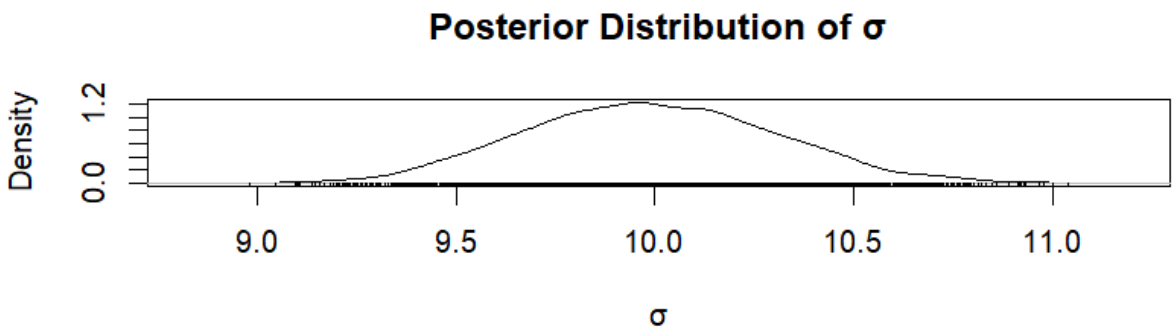
```r
# Gradient functions
gradient <- function(mu, sigma, y, n, m, s, a, b) {
  grad_mu <- (((n * mu) - sum(y)) / (sigma^2)) + ((mu - m) / (s^2))
  grad_sigma <- (n / sigma) - (sum((y - mu)^2) / (sigma^3)) + ((sigma - a) / (b^2))
  return(c(grad_mu, grad_sigma))
}

# Potential energy function
V <- function(mu, sigma, y, n, m, s, a, b) {
  nlpd <- -(sum(dnorm(y, mu, sigma, log = TRUE)) + dnorm(mu, m, s, log = TRUE) + dnorm(sigma, a, b, log = TRUE))
  return(nlpd)
}

# HMC sampler
HMC <- function(y, n, m, s, a, b, step, L, initial_q, nsamp, nburn) {
  mu_chain <- rep(NA, nsamp)
  sigma_chain <- rep(NA, nsamp)
  reject <- 0

  # Initialization of Markov chain
  mu_chain[1] <- initial_q[1]
  sigma_chain[1] <- initial_q[2]

  # Evolution of Markov chain
  i <- 1
  while (i < nsamp) {
    q <- c(mu_chain[i], sigma_chain[i])  # Current position of the particle
    p <- rnorm(length(q), 0, 1)  # Generate random momentum at the current position
    current_q <- q
    current_p <- p
    current_V <- V(current_q[1], current_q[2], y, n, m, s, a, b)  # Current potential energy
    current_T <- sum(current_p^2) / 2  # Current kinetic energy

    # Take L leapfrog steps
    for (l in 1:L) {
      # Change in momentum in 'step/2' time
      p <- p - ((step / 2) * gradient(q[1], q[2], y, n, m, s, a, b))
      # Change in position in 'step' time
      q <- q + step * p
      # Change in momentum in 'step/2' time
      p <- p - ((step / 2) * gradient(q[1], q[2], y, n, m, s, a, b))
    }

    proposed_q <- q
    proposed_p <- p
    proposed_V <- V(proposed_q[1], proposed_q[2], y, n, m, s, a, b)  # Proposed potential energy
    proposed_T <- sum(proposed_p^2) / 2  # Proposed kinetic energy

    accept.prob <- min(1, exp(current_V + current_T - proposed_V - proposed_T))
```

(Top Level)

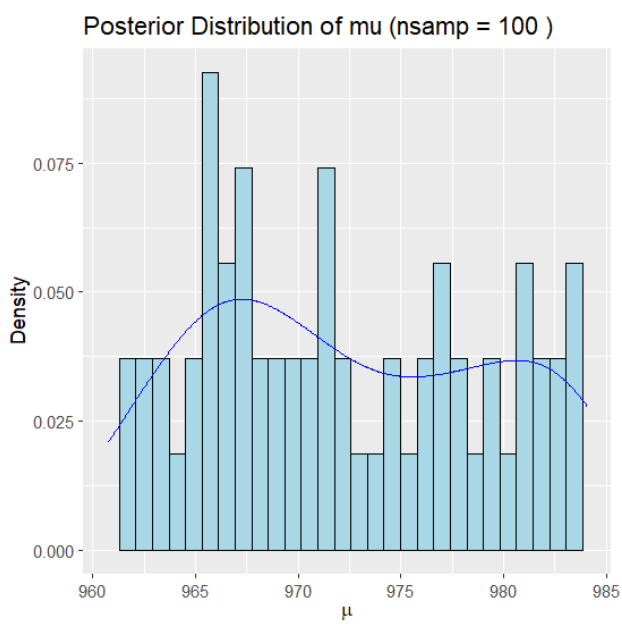
```r
    # Accept/reject the proposed position q
    if (accept.prob > runif(1, 0, 1)) {
      mu_chain[i + 1] <- proposed_q[1]
      sigma_chain[i + 1] <- proposed_q[2]
      i <- i + 1
    } else {
      reject <- reject + 1
    }
  }

  # Create a data frame of posterior samples
  posteriors <- data.frame(mu_chain, sigma_chain)[-c(1:nburn), ]
  posteriors$sample_id <- 1:nrow(posteriors)

  return(posteriors)
}
# Generate data
true_mu <- 800
true_var <- 100
y <- rnorm(500, mean = true_mu, sd = sqrt(true_var))

# Parameters for HMC
nsamp <- 6000
nburn <- 2000
step <- 0.02
L <- 12
initial_q <- c(1000, 11)

# Run HMC sampler
df.posterior <- HMC(y = y, n = length(y), m = 1000, s = 100, a = 10, b = 2,
                    step = step, L = L, initial_q = initial_q,
                    nsamp = nsamp, nburn = nburn)

# Plot posterior distributions
# Plot posterior distributions using density estimation
par(mfrow = c(2, 1))

# Density plot for μ
dens_mu <- density(df.posterior$mu_chain)
plot(dens_mu, main = "Posterior Distribution of μ", xlab = "μ", ylim = c(0, max(dens_mu$y)),
     ylab = "Density")
rug(df.posterior$mu_chain)

# Density plot for σ
dens_sigma <- density(df.posterior$sigma_chain)
plot(dens_sigma, main = "Posterior Distribution of σ", xlab = "σ", ylim = c(0, max(dens_sigma$y)),
     ylab = "Density")
rug(df.posterior$sigma_chain)
```

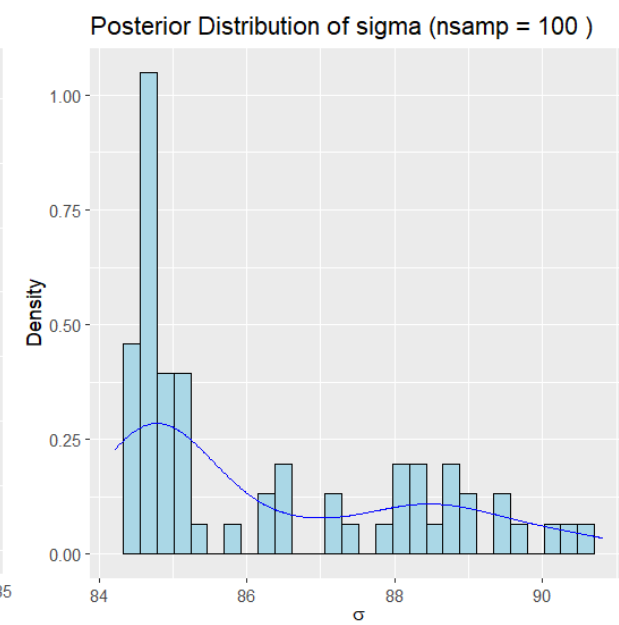
7:1 (Top Level)


Posterior Distribution of μ
Density
0.0
0.8
798
799
800
801
μ


Posterior Distribution of σ
Density
0.0
1.2
9.0
9.5
10.0
10.5
11.0
σ

3.2


Posterior Distribution of mu (nsamp = 100)
Density
0.000
0.025
0.050
0.075
960
965
970
975
980
985
μ


Posterior Distribution of sigma (nsamp = 100)
Density
0.00
0.25
0.50
0.75
1.00
84
86
88
90
σ

code::

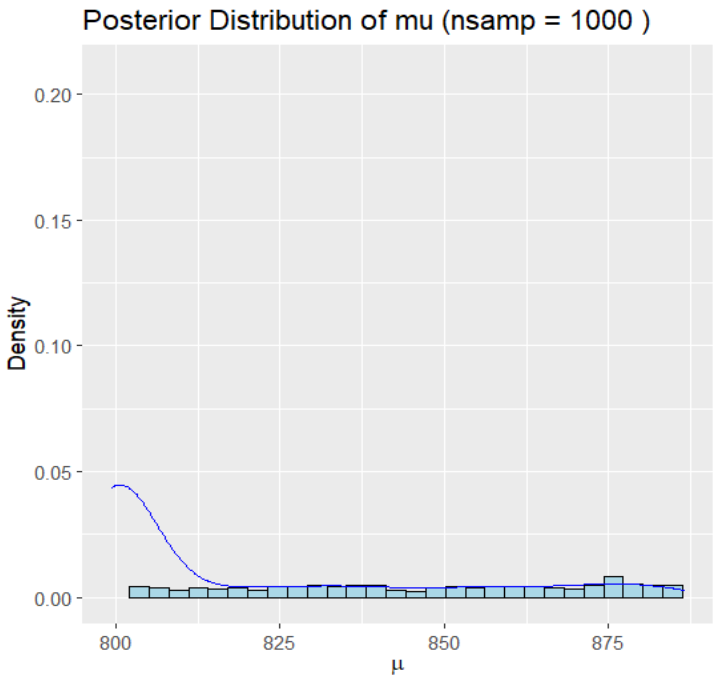
```r

true_mu <- 800
true_var <- 100
y <- rnorm(500, mean = true_mu, sd = sqrt(true_var))
hist(y)

# Gradient function
gradient <- function(mu, sigma, y, n, m, s, a, b) {
  # Compute gradients for mu and sigma
  grad_mu <- (((n * mu) - sum(y)) / (sigma^2)) + ((mu - m) / (s^2))
  grad_sigma <- (n / sigma) - (sum((y - mu)^2) / (sigma^3)) + ((sigma - a) / (b^2))
  return(c(grad_mu, grad_sigma))
}

# Potential energy function
V <- function(mu, sigma, y, n, m, s, a, b) {
  # Compute potential energy based on data likelihood and priors
  nlpd <- -(sum(dnorm(y, mu, sigma, log = TRUE)) + dnorm(mu, m, s, log = TRUE) + dnorm(sigma, a, b, log = TRUE))
  return(nlpd)
}

# HMC sampler function
HMC <- function(y, n, m, s, a, b, step, L, initial_q, nsamp, nburn) {
  # Initialize chains and rejection count
  mu_chain <- rep(NA, nsamp)
  sigma_chain <- rep(NA, nsamp)
  reject <- 0

  # Initialize first position in Markov chain
  mu_chain[1] <- initial_q[1]
  sigma_chain[1] <- initial_q[2]

  # Main loop for HMC sampling
  i <- 1
  while (i < nsamp) {
    q <- c(mu_chain[i], sigma_chain[i])  # Current position
    p <- rnorm(length(q), 0, 1)  # Generate random momentum

    current_q <- q
    current_p <- p
    current_V <- V(current_q[1], current_q[2], y, n, m, s, a, b)  # Current potential energy
    current_T <- sum(current_p^2) / 2  # Current kinetic energy

    # Leapfrog integration
    for (l in 1:L) {
      p <- p - (step / 2) * gradient(q[1], q[2], y, n, m, s, a, b)  # Half step in momentum
      q <- q + step * p  # Full step in position
      p <- p - (step / 2) * gradient(q[1], q[2], y, n, m, s, a, b)  # Half step in momentum
    }
```

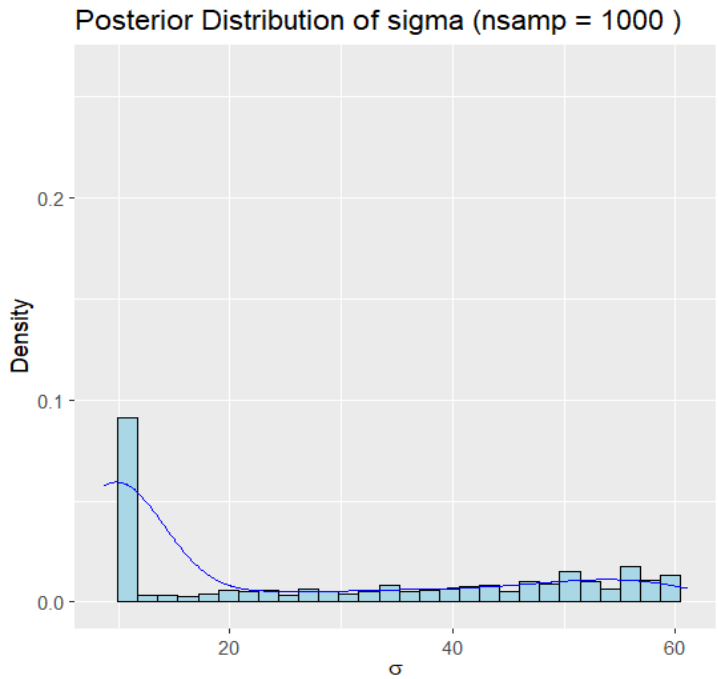
1:1 (Top Level)

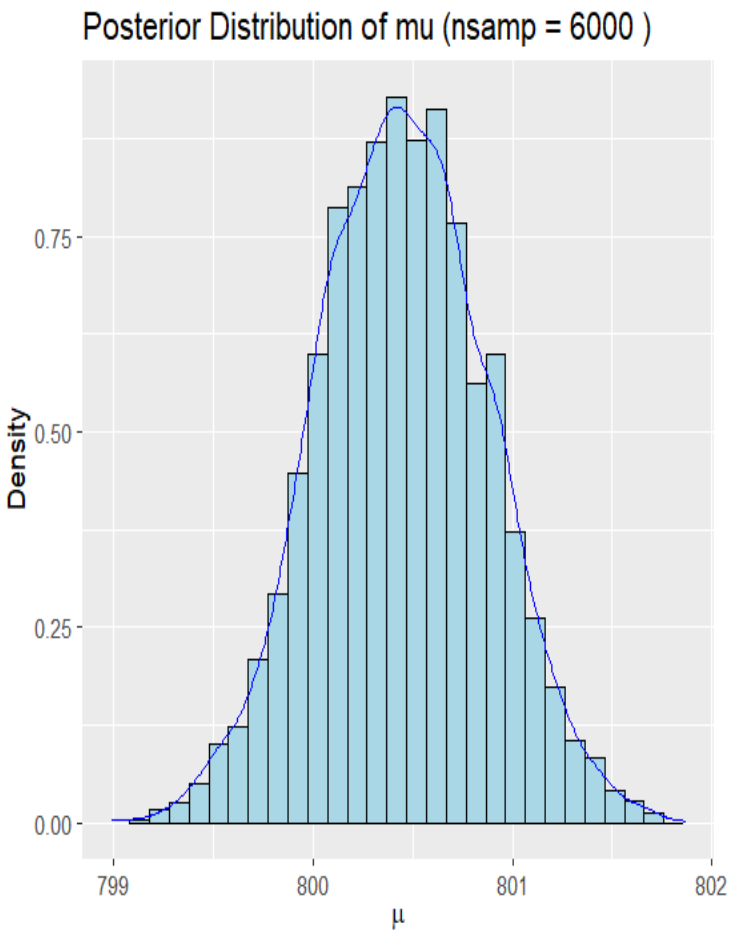
```r

    proposed_q <- q
    proposed_p <- p

    proposed_V <- V(proposed_q[1], proposed_q[2], y, n, m, s, a, b)  # Proposed potential energy
    proposed_T <- sum(proposed_p^2) / 2  # Proposed kinetic energy

    # Metropolis-Hastings acceptance criterion
    accept_prob <- min(1, exp(current_V + current_T - proposed_V - proposed_T))

    # Accept/reject proposal
    if (accept_prob > runif(1)) {
      mu_chain[i + 1] <- proposed_q[1]
      sigma_chain[i + 1] <- proposed_q[2]
      i <- i + 1
    } else {
      reject <- reject + 1
    }
  }

  # Create data frame of posterior samples
  posteriors <- data.frame(mu_chain, sigma_chain)
  posteriors <- posteriors[-c(1:nburn), ]  # Remove burn-in samples

  # Add sample_id column for plotting
  posteriors$sample_id <- 1:nrow(posteriors)

  return(posteriors)
}
# Function to run HMC sampler and plot posterior distributions
run_HMC_and_plot <- function(nsamp, nburn = NULL, step = 0.02, L = 12, initial_q = c(1000, 11)) {
  if (is.null(nburn) || nburn >= nsamp) {
    nburn <- floor(nsamp / 3)
  }

  posterior <- HMC(y = y, n = length(y), m = 1000, s = 20, a = 10, b = 2,
                   step = step, L = L, initial_q = initial_q,
                   nsamp = nsamp, nburn = nburn)

  plot_posteriors(posterior, nsamp)
}

# Function to plot posterior distributions
plot_posteriors <- function(posterior, nsamp) {
  library(ggplot2)
```

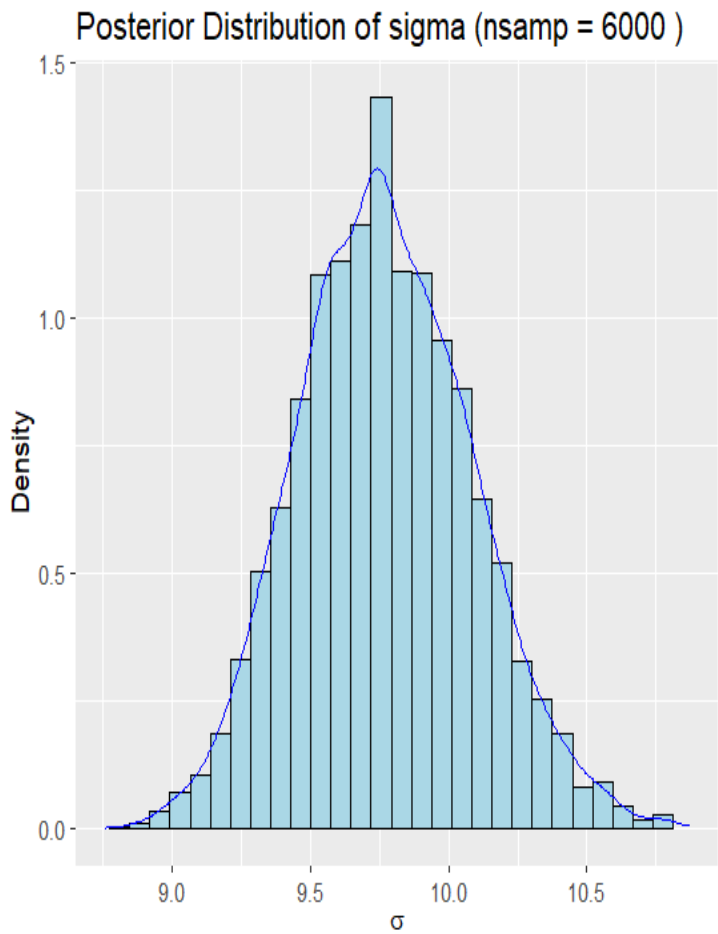
```r
# Function to run HMC sampler and plot posterior distributions
run_HMC_and_plot <- function(nsamp, nburn = NULL, step = 0.02, L = 12, initial_q = c(1000, 11)) {
  if (is.null(nburn) || nburn >= nsamp) {
    nburn <- floor(nsamp / 3)
  }

  posterior <- HMC(y = y, n = length(y), m = 1000, s = 20, a = 10, b = 2,
                   step = step, L = L, initial_q = initial_q,
                   nsamp = nsamp, nburn = nburn)

  plot_posteriors(posterior, nsamp)
}

# Function to plot posterior distributions
plot_posteriors <- function(posterior, nsamp) {
  library(ggplot2)

  # Plot for mu
  p1 <- ggplot(posterior, aes(x = mu_chain)) +
    geom_histogram(aes(y = ..density..), bins = 30, color = "black", fill = "lightblue") +
    geom_density(color = "blue") +
    labs(title = paste("Posterior Distribution of mu (nsamp =", nsamp, ")"),
         x = expression(mu), y = "Density") +
    xlim(range(posterior$mu_chain))  # Adjust x-axis limits

  # Plot for sigma
  p2 <- ggplot(posterior, aes(x = sigma_chain)) +
    geom_histogram(aes(y = ..density..), bins = 30, color = "black", fill = "lightblue") +
    geom_density(color = "blue") +
    labs(title = paste("Posterior Distribution of sigma (nsamp =", nsamp, ")"),
         x = expression(sigma), y = "Density") +
    xlim(range(posterior$sigma_chain))  # Adjust x-axis limits

  return(list(p1, p2))
}

# Running HMC and plotting for different nsamp values
plots_100 <- run_HMC_and_plot(nsamp = 100)
plots_1000 <- run_HMC_and_plot(nsamp = 1000)
plots_6000 <- run_HMC_and_plot(nsamp = 6000)

# Displaying plots using grid.arrange from gridExtra package
library(gridExtra)
grid.arrange(grobs = plots_100, ncol = 2)
grid.arrange(grobs = plots_1000, ncol = 2)
grid.arrange(grobs = plots_6000, ncol = 2)
```

1:1 (Top Level)


Posterior Distribution of mu (nsamp = 1000)
Density
0.20
0.15
0.10
0.05
0.00
800
825
850
875
μ


Posterior Distribution of sigma (nsamp = 1000)
Density
0.2
0.1
0.0
20
40
60
σ


Posterior Distribution of mu (nsamp = 6000)
Density
0.75
0.50
0.25
0.00
799
800
801
802
μ


Posterior Distribution of sigma (nsamp = 6000)
Density
1.5
1.0
0.5
0.0
9.0
9.5
10.0
10.5
σ

3.3

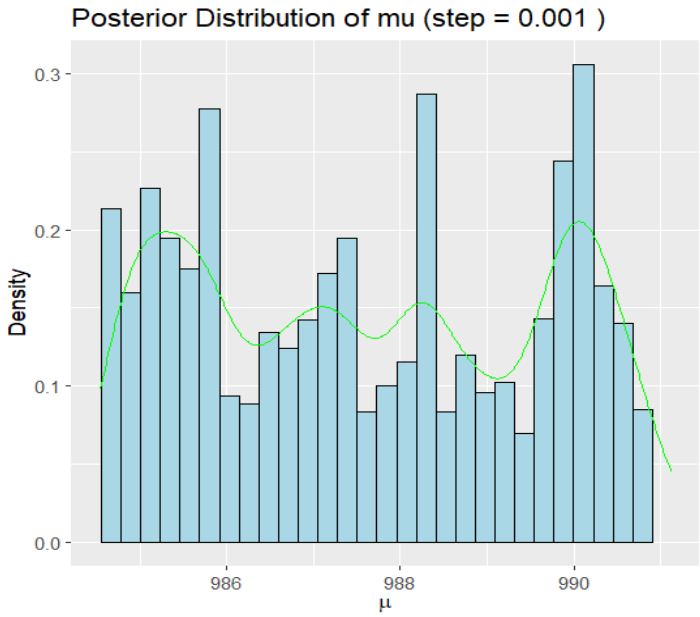
RStudio Source Editor

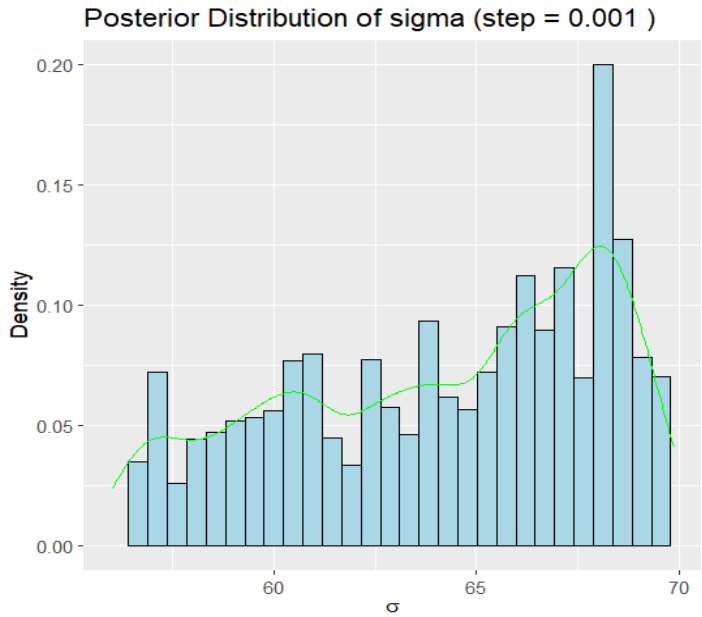
Untitled7*

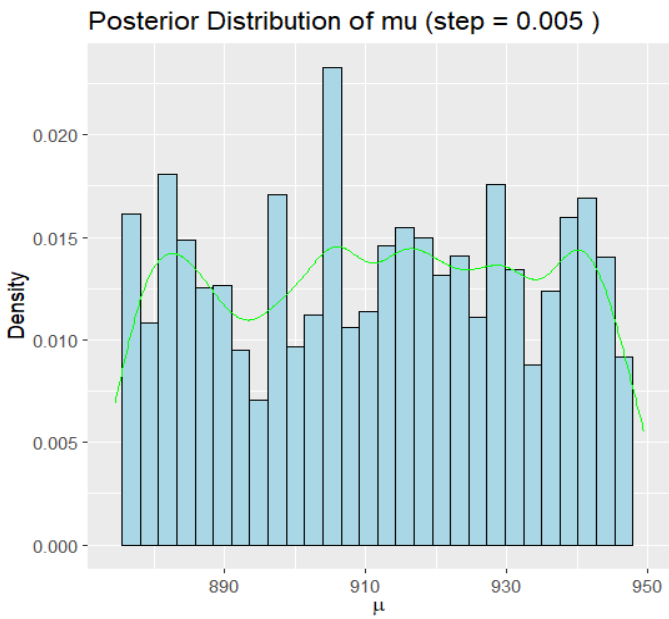
```r
run_HMC_and_plot <- function(nsamp, step_sizes, nburn = NULL, L = 12, initial_q = c(1000, 11)) {
  if (is.null(nburn) || nburn >= nsamp) {
    nburn <- floor(nsamp / 3)
  }

  posterior_list <- list()

  for (step in step_sizes) {
    posterior <- HMC(y = y, n = length(y), m = 1000, s = 20, a = 10, b = 2,
                     step = step, L = L, initial_q = initial_q,
                     nsamp = nsamp, nburn = nburn)

    posterior_list[[as.character(step)]] <- posterior
  }

  plot_list <- list()

  for (step in step_sizes) {
    plot_list[[as.character(step)]] <- plot_posteriors(posterior_list[[as.character(step)]], step)
  }

  return(plot_list)
}

# Function to plot posterior distributions
plot_posteriors <- function(posterior, step) {
  library(ggplot2)

  # Plot for mu
  p1 <- ggplot(posterior, aes(x = mu_chain)) +
    geom_histogram(aes(y = ..density..), bins = 30, color = "black", fill = "lightblue") +
    geom_density(color = "green") +
    labs(title = paste("Posterior Distribution of mu (step =", step, ")"),
         x = expression(mu), y = "Density") +
    xlim(range(posterior$mu_chain))  # Adjust x-axis limits

  # Plot for sigma
  p2 <- ggplot(posterior, aes(x = sigma_chain)) +
    geom_histogram(aes(y = ..density..), bins = 30, color = "black", fill = "lightblue") +
    geom_density(color = "green") +
    labs(title = paste("Posterior Distribution of sigma (step =", step, ")"),
         x = expression(sigma), y = "Density") +
    xlim(range(posterior$sigma_chain))  # Adjust x-axis limits

  return(list(p1, p2))
```

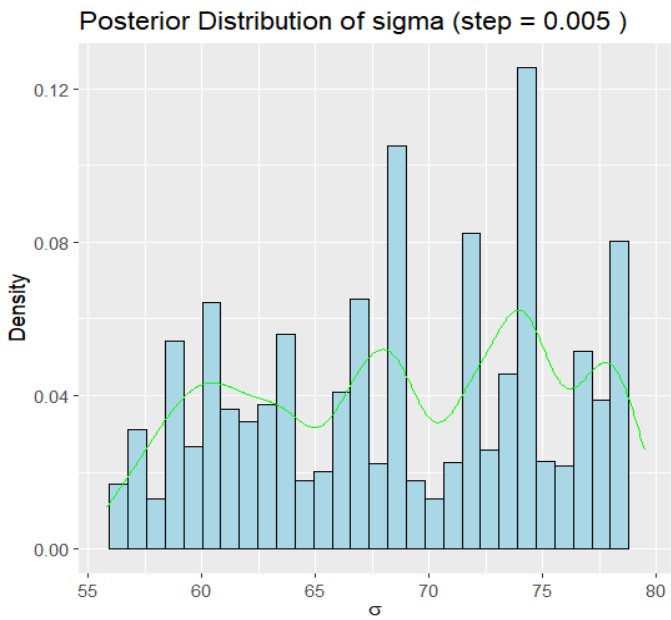
48:39 (Top Level) R Script

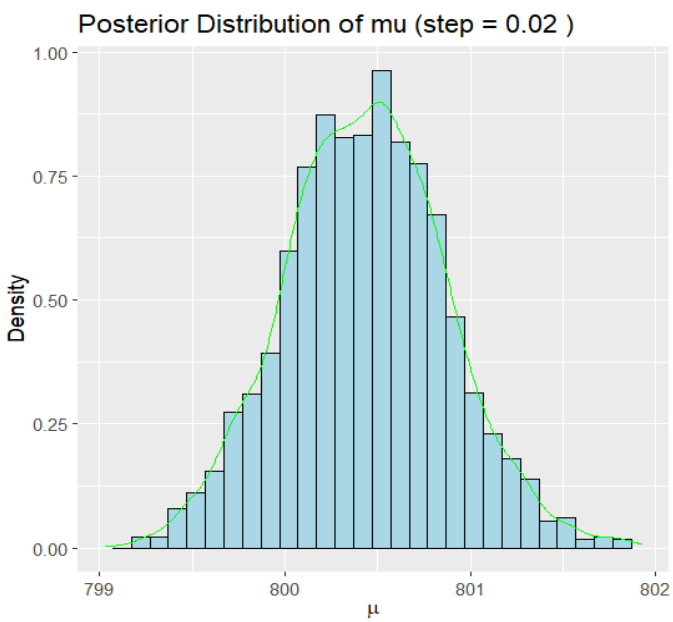
```r
step_sizes <- c(0.001, 0.005, 0.02)
plots <- run_HMC_and_plot(nsamp = 6000, step_sizes=
 step_sizes)
library(gridExtra)

grid.arrange(grobs = plots[[1]], ncol = 2)
grid.arrange(grobs = plots[[2]], ncol = 2)
grid.arrange(grobs = plots[[3]], ncol = 2)
```

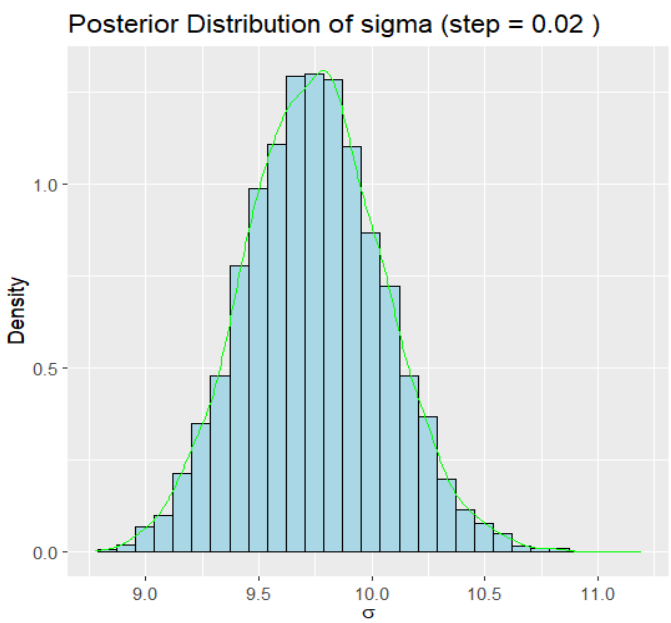

Posterior Distribution of mu (step = 0.001)
Density
0.0
0.1
0.2
0.3
986
988
990
μ


Posterior Distribution of sigma (step = 0.001)
Density
0.00
0.05
0.10
0.15
0.20
60
65
70
σ


Posterior Distribution of mu (step = 0.005)
Density
0.000
0.005
0.010
0.015
0.020
890
910
930
950
μ


Posterior Distribution of sigma (step = 0.005)
Density
0.00
0.04
0.08
0.12
55
60
65
70
75
80
σ


Posterior Distribution of mu (step = 0.02)
Density
0.00
0.25
0.50
0.75
1.00
799
800
801
802
μ


Posterior Distribution of sigma (step = 0.02)
Density
0.0
0.5
1.0
9.0
9.5
10.0
10.5
11.0
σ

3.4:--

For step size=0.001:: The posterior distribution of mu exhibits characteristics of poor mixing and insufficient exploration of the parameter space. The plot shows irregularities and lacks a clear, well-defined peak, suggesting that the step size of 0.001 may not be adequate for effective sampling.

For step size=0.02:: In contrast, the posterior distribution of mu with a step size of 0.02 demonstrates a bell-shaped and symmetric plot. The distribution shows a clear peak, indicating effective sampling and proper exploration of the parameter space.

3.5:--

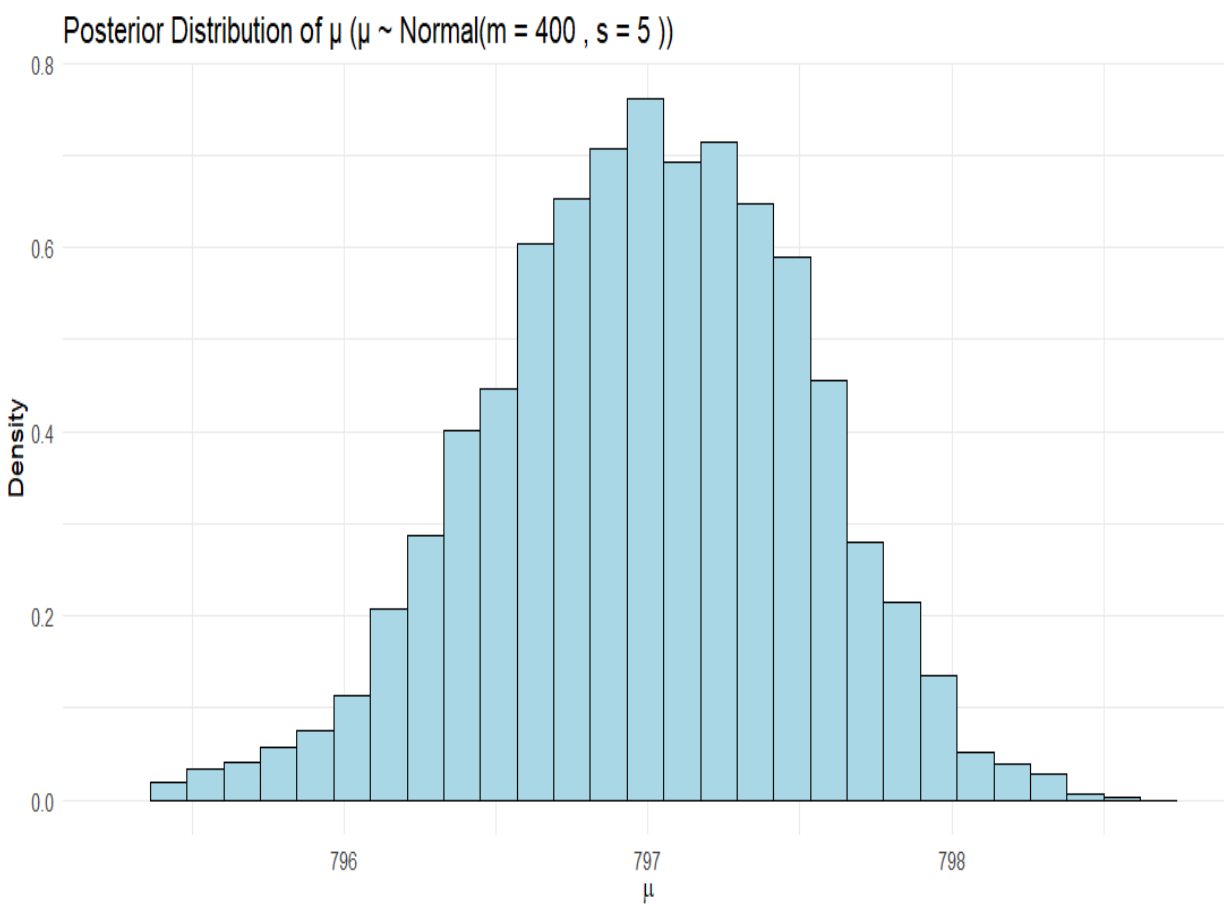
RStudio Source Editor

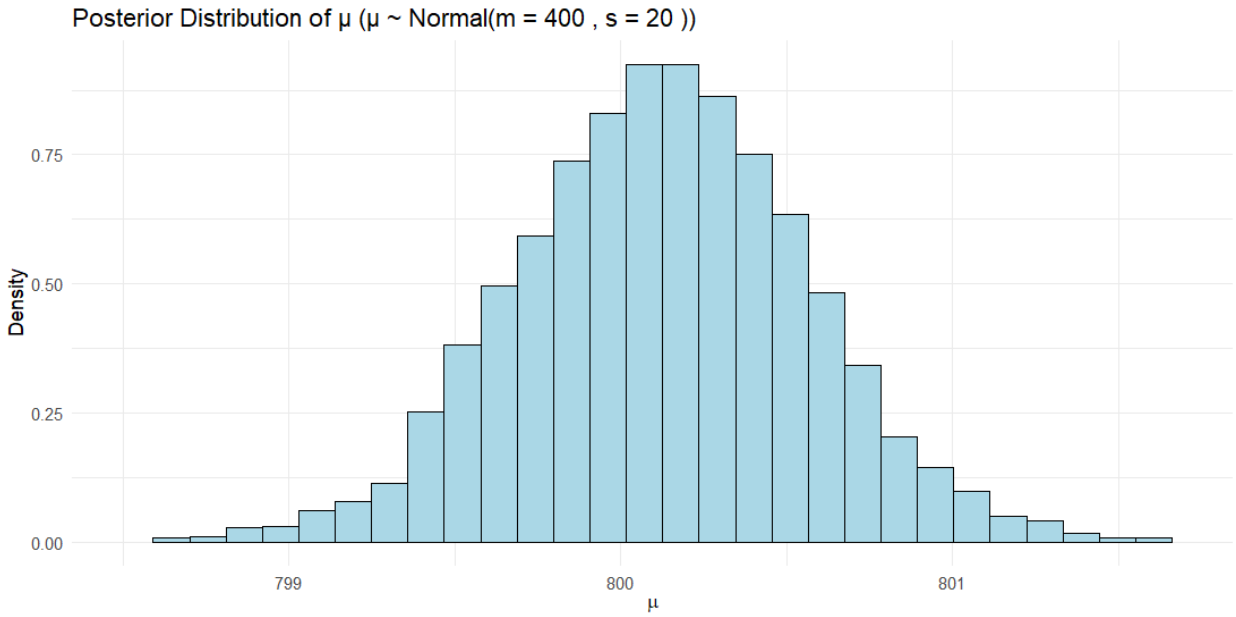
Untitled9*

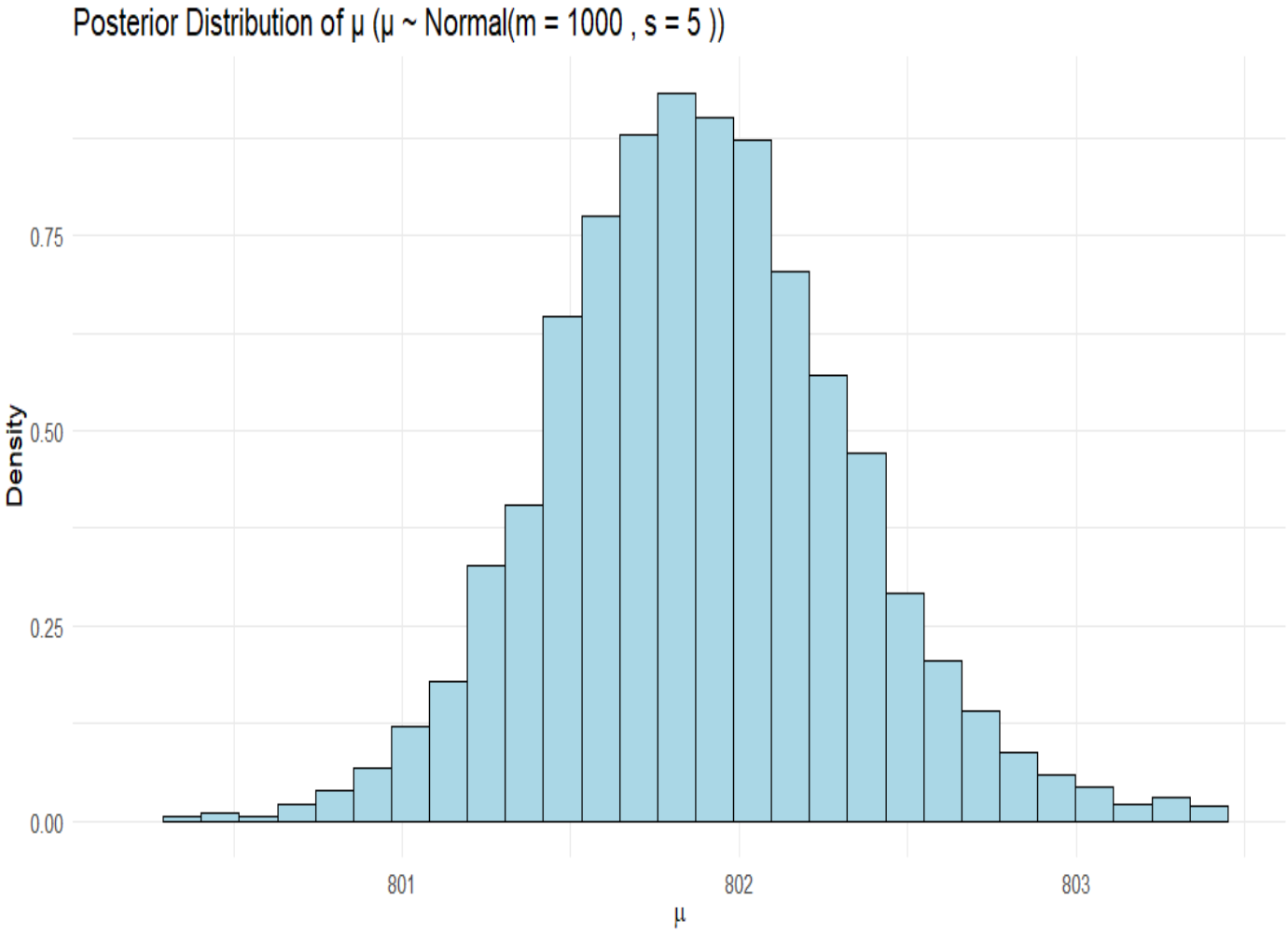
```r
# Function to run HMC sampler with different priors on μ and return posterior data
run_HMC_with_prior <- function(prior_mean, prior_sd, nsamp = 6000, nburn = 2000,
                               step = 0.02, L = 12, initial_q = c(1000, 11)) {

  # Define prior parameters
  m <- prior_mean
  s <- prior_sd

  # Run HMC sampler
  posterior <- HMC(y = y, n = length(y), m = m, s = s, a = 10, b = 2,
                   step = step, L = L, initial_q = initial_q,
                   nsamp = nsamp, nburn = nburn)

  # Return posterior data frame
  return(posterior)
}

# Generate data y
set.seed(123)
true_mu <- 800
true_var <- 100
y <- rnorm(500, mean = true_mu, sd = sqrt(true_var))

# Define different priors for μ
prior_scenarios <- list(
  list(mean = 400, sd = 5),
  list(mean = 400, sd = 20),
  list(mean = 1000, sd = 5),
  list(mean = 1000, sd = 20),
  list(mean = 1000, sd = 100)
)

# Run HMC sampler for each prior scenario
posteriors <- list()

for (scenario in prior_scenarios) {
  prior_mean <- scenario$mean
  prior_sd <- scenario$sd

  posterior <- run_HMC_with_prior(prior_mean, prior_sd)
  posteriors[[paste("μ ~ Normal(m =", prior_mean, ", s =", prior_sd, ")")]] <- posterior
}

# Plot posterior distributions for μ
library(ggplot2)

plot_posteriors <- function(posterior_list) {
  plots <- list()

```

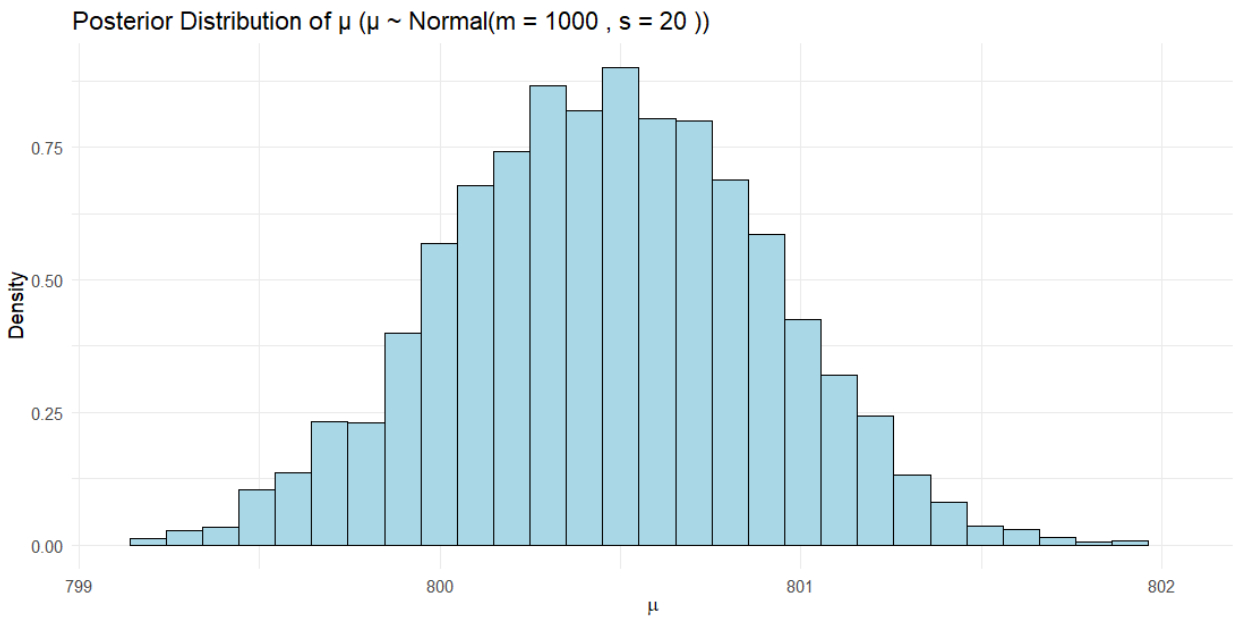
62:3 plot_posteriors(posterior_list) R Script

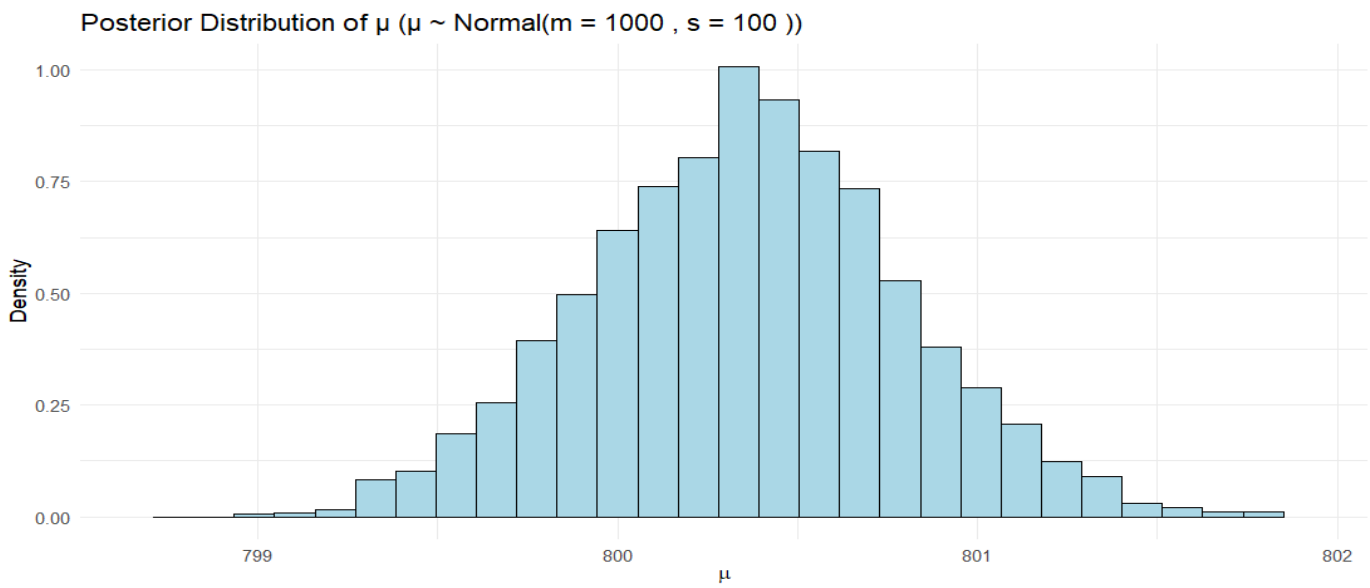
```r
  for (key in names(posterior_list)) {
    posterior <- posterior_list[[key]]

    p <- ggplot(posterior, aes(x = mu_chain)) +
      geom_histogram(aes(y = ..density..), bins = 30, color = "black", fill = "lightblue") +
      geom_density(color = "blue") +
      labs(title = paste("Posterior Distribution of μ (", key, ")"),
           x = expression(mu), y = "Density") +
      xlim(range(posterior$mu_chain))  # Adjust x-axis limits

    plots[[key]] <- p
  }

  return(plots)
}

# Plotting posterior distributions for μ
plots <- plot_posteriors(posteriors)

# Display plots using grid.arrange from gridExtra package
library(gridExtra)
grid.arrange(grobs = plots, ncol = 2)
```




Posterior Distribution of μ (μ ~ Normal(m = 400 , s = 20))
Density
0.75
0.50
0.25
0.00
799
800
801
μ


Posterior Distribution of μ (μ ~ Normal(m = 1000 , s = 5))
Density
0.75
0.50
0.25
0.00
801
802
803
μ





****

PART-2::-Writing sampler for Bayesian inference

****

2.1

RStudio Source Editor

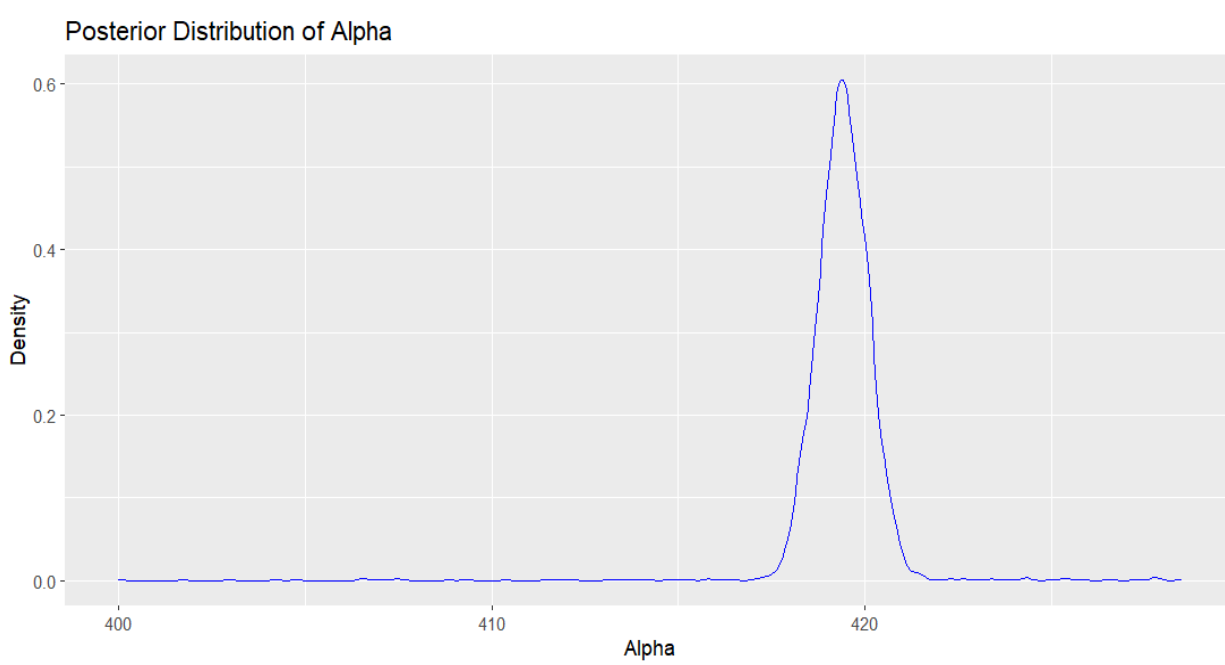
Untitled2*

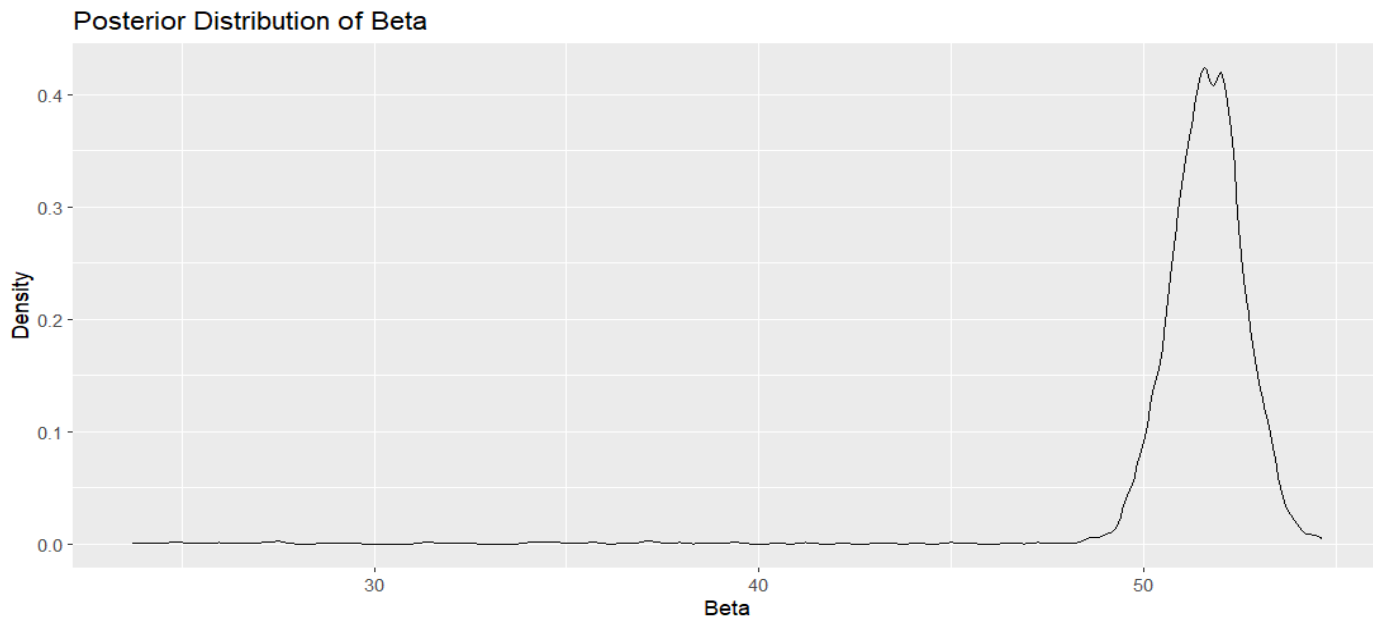
Source on Save

```r
# Load required libraries
library(truncnorm)
library(ggplot2)

# Function to calculate the log-likelihood
log_likelihood <- function(data, alpha, beta, sigma) {
  mu <- alpha + beta * data$type
  sum(dnorm(data$RT, mean = mu, sd = sigma, log = TRUE))
}

# Function to calculate the log-prior
log_prior <- function(alpha, beta) {
  log_prior_alpha <- dnorm(alpha, mean = 400, sd = 50, log = TRUE)
  log_prior_beta <- log(dtruncnorm(beta, a = 0, b = Inf, mean = 0, sd = 50))
  log_prior_alpha + log_prior_beta
}

# Function to calculate the log-posterior
log_posterior <- function(data, alpha, beta, sigma) {
  log_likelihood(data, alpha, beta, sigma) + log_prior(alpha, beta)
}

# Metropolis-Hastings sampler
metropolis_hastings <- function(data, iter = 10000, sigma_alpha = 1, sigma_beta = 1, sigma = 30) {
  alpha <- 400
  beta <- 25
  samples <- matrix(NA, nrow = iter, ncol = 2)
  colnames(samples) <- c("alpha", "beta")

  current_log_post <- log_posterior(data, alpha, beta, sigma)

  for (i in 1:iter) {
    # Propose new values
    alpha_new <- rnorm(1, mean = alpha, sd = sigma_alpha)
    beta_new <- rtruncnorm(1, a = 0, b = Inf, mean = beta, sd = sigma_beta)

    # Calculate new log-posterior
    new_log_post <- log_posterior(data, alpha_new, beta_new, sigma)

    # Acceptance probability
    accept_prob <- exp(new_log_post - current_log_post)

    if (runif(1) < accept_prob) {
      alpha <- alpha_new
      beta <- beta_new
      current_log_post <- new_log_post
    }

    samples[i, ] <- c(alpha, beta)
```

16:2 (Top Level)

```r
    # Calculate new log-posterior
    new_log_post <- log_posterior(data, alpha_new, beta_new, sigma)

    # Acceptance probability
    accept_prob <- exp(new_log_post - current_log_post)

    if (runif(1) < accept_prob) {
      alpha <- alpha_new
      beta <- beta_new
      current_log_post <- new_log_post
    }

    samples[i, ] <- c(alpha, beta)
  }

  as.data.frame(samples)
}

# Load the data
dat <- read.table(
  "https://raw.githubusercontent.com/yadavhimanshu059/CGS698C/main/notes/Data/word-recognition-times.csv",
  sep = ",", header = TRUE
)[, -1]

# Convert 'type' column to binary
dat$type <- ifelse(dat$type == "non-word", 1, 0)

# Run the Metropolis-Hastings sampler
set.seed(123)  # For reproducibility
samples <- metropolis_hastings(dat)

# Plot the posterior distributions
ggplot(samples, aes(x = alpha)) +
  geom_density(color = "blue") + xlim(410,430)
  labs(title ="Posterior Distribution of Alpha", x = "Alpha", y = "Density")

ggplot(samples, aes(x = beta)) +
  geom_density(color = "black") + xlim(45,55)
  labs(title = "Posterior Distribution of Beta", x = "Beta", y = "Density")

# Calculate 95% credible intervals
credible_intervals <- lapply(samples, function(x) quantile(x, c(0.025, 0.975)))

# Display the 95% credible intervals
cat("95% Credible Intervals:\n")
cat(paste("Alpha:", round(credible_intervals$alpha, 2), "\n"))
cat(paste("Beta:", round(credible_intervals$beta, 2), "\n"))
```





2.2.

```
95% Credible Intervals:

> cat(paste("Alpha:", round(credible_intervals$alpha, 2), "\n"))
Alpha: 418.09
Alpha: 420.81
> cat(paste("Beta:", round(credible_intervals$beta, 2), "\n"))
Beta: 49.55
Beta: 53.44
```