MTH422_Assignment-3

Charitha

2024-03-11

Assignment - 3

Question - 1

(a)

Jeffreys' prior distribution and report the posterior 95% credible set for each county.

we have considered the approve as no of success and (approve + disapprove) as total numbers, it means we can assume it may be **Binomial distribution**. Then the Jeffreys' prior distribution is **Beta(0.5,0.5)**.

```
set.seed(123)
approve <- c(12,90,80,5,63,15,67,22,56,33)
disapprove <- c(50,150,63,10,63,8,56,19,63,19)
county <- 1:10

jeffreys_CI <- function(p,q)
{
  n <- p + q
  beta_lower <- qbeta(0.025, p + 0.5, q + 0.5)
  beta_upper <- qbeta(0.975, p + 0.5, q + 0.5)
  return(c(beta_lower * n, beta_upper * n))
}

for (i in 1:length(approve))
{
  p <- approve[i]
  q <- disapprove[i]
  CI <- jeffreys_CI(p,q)
  cat("County", county[i], ":\n")
  cat("Posterior 95% Credible Interval: (", CI[1], ", ", CI[2], ")\n\n")
}
```

```
## County 1 :
## Posterior 95% Credible Interval: ( 6.8444 ,  18.8809 )
##
## County 2 :
## Posterior 95% Credible Interval: ( 75.73074 ,  104.9879 )
##
## County 3 :
## Posterior 95% Credible Interval: ( 68.29104 ,  91.36875 )
##
## County 4 :
## Posterior 95% Credible Interval: ( 2.10379 ,  8.762457 )
##
## County 5 :
## Posterior 95% Credible Interval: ( 52.10441 ,  73.89559 )
##
## County 6 :
## Posterior 95% Credible Interval: ( 10.32416 ,  18.85488 )
##
## County 7 :
## Posterior 95% Credible Interval: ( 56.15233 ,  77.5922 )
##
## County 8 :
## Posterior 95% Credible Interval: ( 15.81904 ,  27.97733 )
##
## County 9 :
## Posterior 95% Credible Interval: ( 45.51942 ,  66.64854 )
##
## County 10 :
## Posterior 95% Credible Interval: ( 25.96488 ,  39.27984 )
```

(b)

For finding a and b,

```
set.seed(123)
prop <- approve/(approve + disapprove)

#sample proportions
smean <- mean(prop)
svariance <- var(prop)

# Solve for a and b
a <- smean * ((smean * (1 - smean)) / svariance - 1)
b <- (1 - smean) * ((smean * (1 - smean)) / svariance - 1)

a
```

```
## [1] 5.433856
```

```
b
```

```
## [1] 5.886676
```

(c)

Empirical Bayesian analysis, for above after finding a and b. The prior is **Beta(a,b)** and Likelihood is **Binomial(n,p)** then the posterior is **Beta(y+a,n-y+b)**.

```
set.seed(100)
empirical_CI <- function(p, q, a ,b)
{
  n <- p + q
  post_a <- p + a
  post_b <- q + b
  beta_lower <- qbeta(0.025, post_a, post_b)
  beta_upper <- qbeta(0.975, post_a, post_b)
  return(c(beta_lower * n, beta_upper * n))
}

posterior_credible_sets <- list()

for (i in 1:length(approve))
{
  p <- approve[i]
  q <- disapprove[i]
  CI <- empirical_CI(p, q, a, b)
  posterior_credible_sets[[paste0("County", county[i])]] <- CI
  cat("County", county[i], ":\n")
  cat("Posterior 95% Credible Interval: (", CI[1], ", ", CI[2],")\n\n")
}
```

```
## County 1 :
## Posterior 95% Credible Interval: ( 9.189232 ,  21.13428 )
##
## County 2 :
## Posterior 95% Credible Interval: ( 76.99429 ,  105.7111 )
##
## County 3 :
## Posterior 95% Credible Interval: ( 67.90862 ,  90.23643 )
##
## County 4 :
## Posterior 95% Credible Interval: ( 3.332125 ,  8.784073 )
##
## County 5 :
## Posterior 95% Credible Interval: ( 52.31259 ,  73.27765 )
##
## County 6 :
## Posterior 95% Credible Interval: ( 9.875375 ,  17.26991 )
##
## County 7 :
## Posterior 95% Credible Interval: ( 55.94767 ,  76.57434 )
##
## County 8 :
## Posterior 95% Credible Interval: ( 15.98908 ,  26.93464 )
##
## County 9 :
## Posterior 95% Credible Interval: ( 46.00369 ,  66.28985 )
##
## County 10 :
## Posterior 95% Credible Interval: ( 25.21219 ,  37.58002 )
```

(d)

- **Jeffreys' Prior Analysis:**
  - Advantages:
    - The Jeffreys' prior is considered non-informative and can be used when no prior information is available.
    - Conceptually simple and easy to implement.
  - Disadvantages:
    - Does not incorporate any information from the observed data, which might lead to wider posterior credible intervals.
    - Might not be appropriate when there is relevant prior information available.
- **Empirical Bayesian Analysis:**
  - Advantages:
    - Incorporates some information from the observed data by estimating the prior distribution, potentially leading to narrower posterior credible intervals.
    - Provides a systematic way to incorporate data-driven prior information.
  - Disadvantages:

- Relies on the method of moments to estimate the prior distribution, which might not capture the true underlying prior distribution accurately, especially with limited data.
- Results can be sensitive to the choice of prior estimation method.

Question - 2

(a)

MAP estimator for μ.

$$\begin{aligned}\hat{\mu}_{MAP} &= \arg\max_{\mu}\{\log(f(\mu|\mathbf{Y})) + \log(\pi(\mu))\} \\ &= \arg\max_{\mu}\log(f(\mu|\mathbf{Y})) \\ &= \arg\max_{\mu}\log\{\Pi_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma_i}\exp[\frac{-(Y_i-\mu)^2}{2\sigma_i^2}]\}\end{aligned}$$

(b)

We observe $n = 3, Y_1 = 12, Y_2 = 10, Y_3 = 22, \sigma_1 = \sigma_2 = 3 and \sigma_3 = 10$, the MAP estimate of μ

```
Y <- c(12,10,22)
sigma <- c(3,3,10)
sigma2 <- sigma^2
post_var <- 1/sum(1/(sigma2))
map_mean <- sum(Y/(sigma2)) * post_var
map_mean
```

```
## [1] 11.47368
```

(c)

Numerical integration to compute posterior mean of μ.

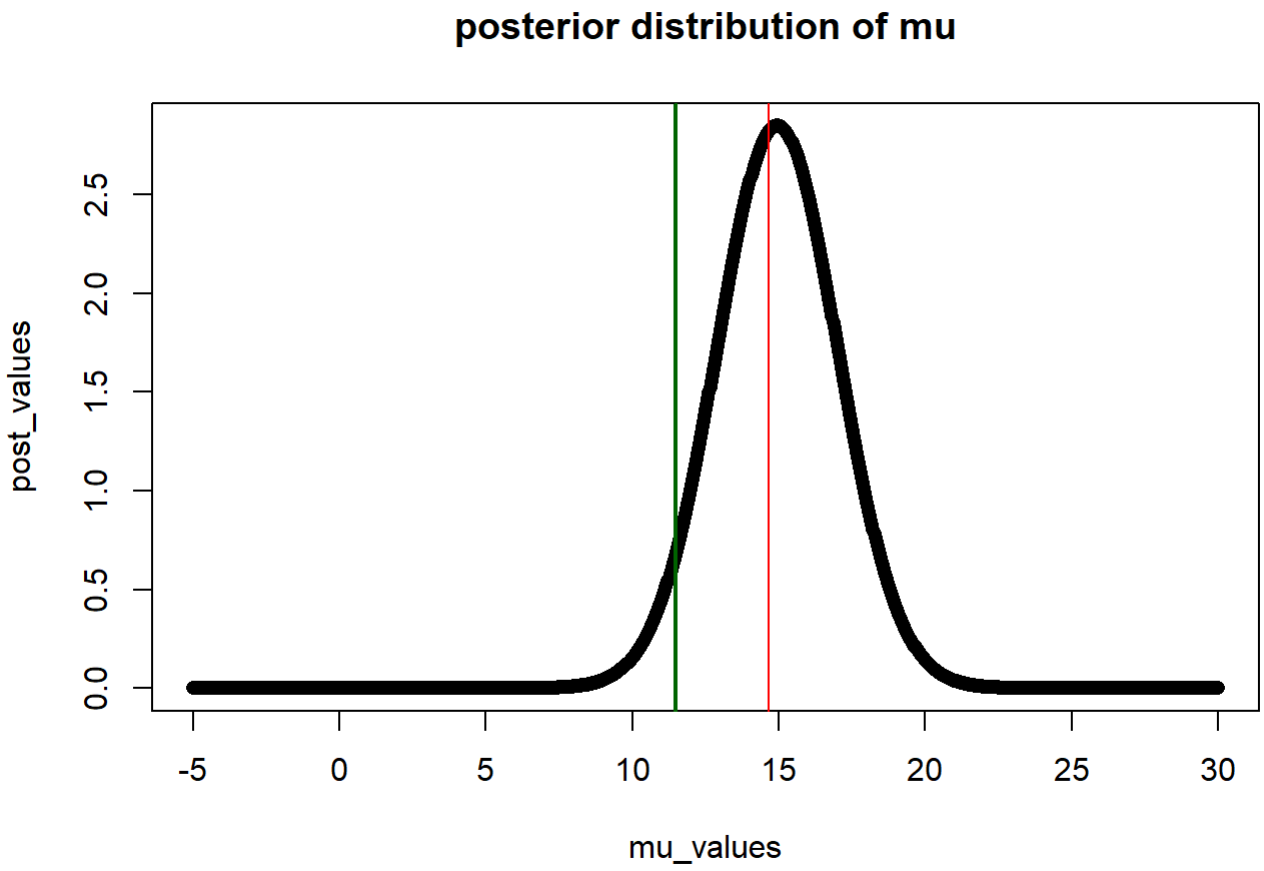
```
posterior_distribution <- function(mu){
  mu * dnorm(mu, mean=mean(Y), sd=sqrt(post_var))
}

post_mean_numerical <- integrate(posterior_distribution, -Inf, +Inf)$value
post_mean_numerical
```

```
## [1] 14.66667
```

(d)

Graph of the posterior distribution of μ.

```
mu_values <- seq(-5,30,length.out=1e3)
post_values <- sapply(mu_values,posterior_distribution)
plot(mu_values,post_values,pch=16,main = "posterior distribution of mu")
abline(v=map_mean,col="darkgreen",lwd=2)
abline(v=post_mean_numerical,col="red")
```



Question - 3

(a)

- Full conditional posterior distributions for σ_1^2 is Inverse Gamma($a + \frac{1}{2}, \frac{Y_1^2}{2} + b$).
- Full conditional posterior distributions for b is Exponential($\frac{1}{\sigma_1^2} + 1$).

(b)

Pseudo code for Gibbs Sampling

- **Step-1:** Set initial values $(\sigma_1^{2\,(0)}, b^{(0)})$
- **Step-2:** Updating
  - For iteration t,
    - FC1 : Draw $\sigma_1^{2\,(t)} | b^{(t-1)}, \mathbf{Y}$
    - FC2 : Draw $b^{(t)} | \sigma_1^{2\,(t)}, \mathbf{Y}$

  We repeat step B times ang get the posterior draws

$$\sigma_1^{2\,(1)}, \sigma_1^{2\,(2)}, \ldots\ldots \sigma_1^{2\,(B)}$$
$$b^{(1)}, b^{(2)}, \ldots\ldots, b^{(B)}$$

(c)

Gibbs Sampling code.

Assume $n = 10, a = 10$ and $Y_i = i$ for $i = 1, 2, \ldots .10$

```
set.seed(123)
library(MCMCpack)
```

```
## Loading required package: coda
```

```
## Loading required package: MASS
```

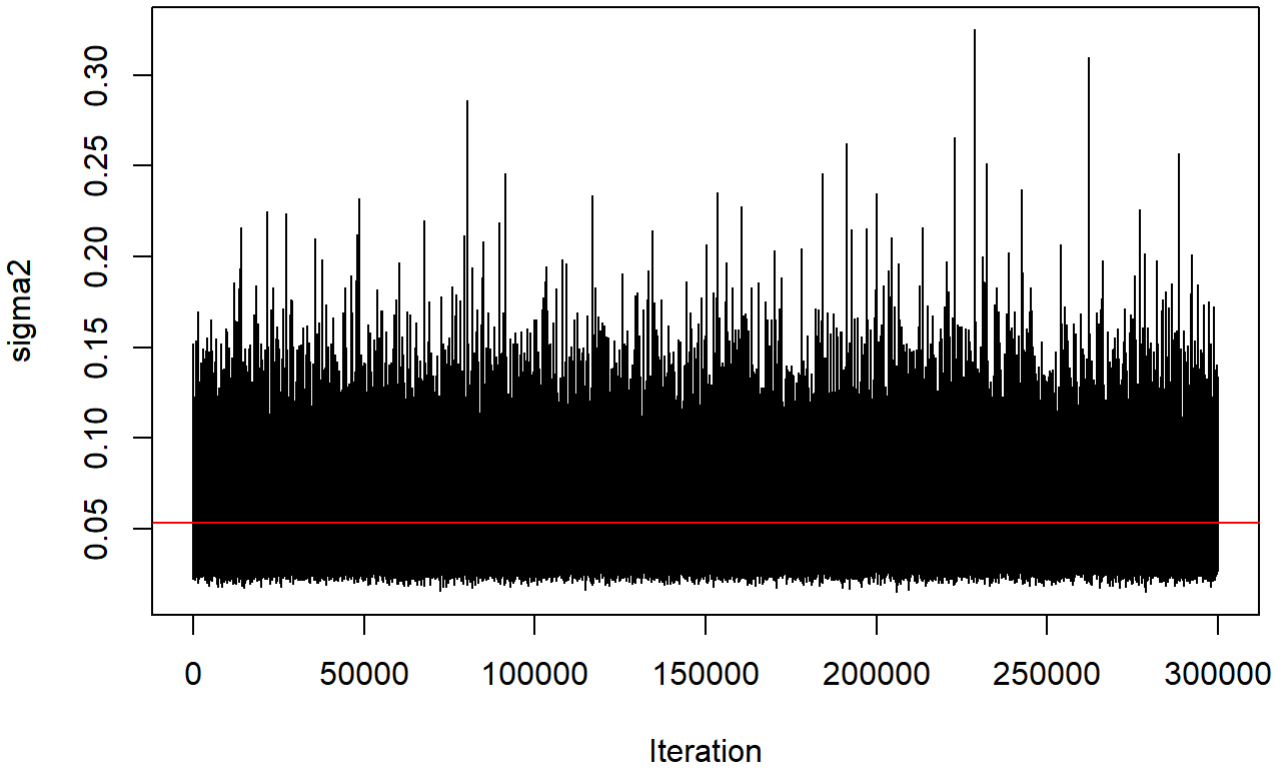
```
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
```

```
## ## Copyright (C) 2003-2024 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
```

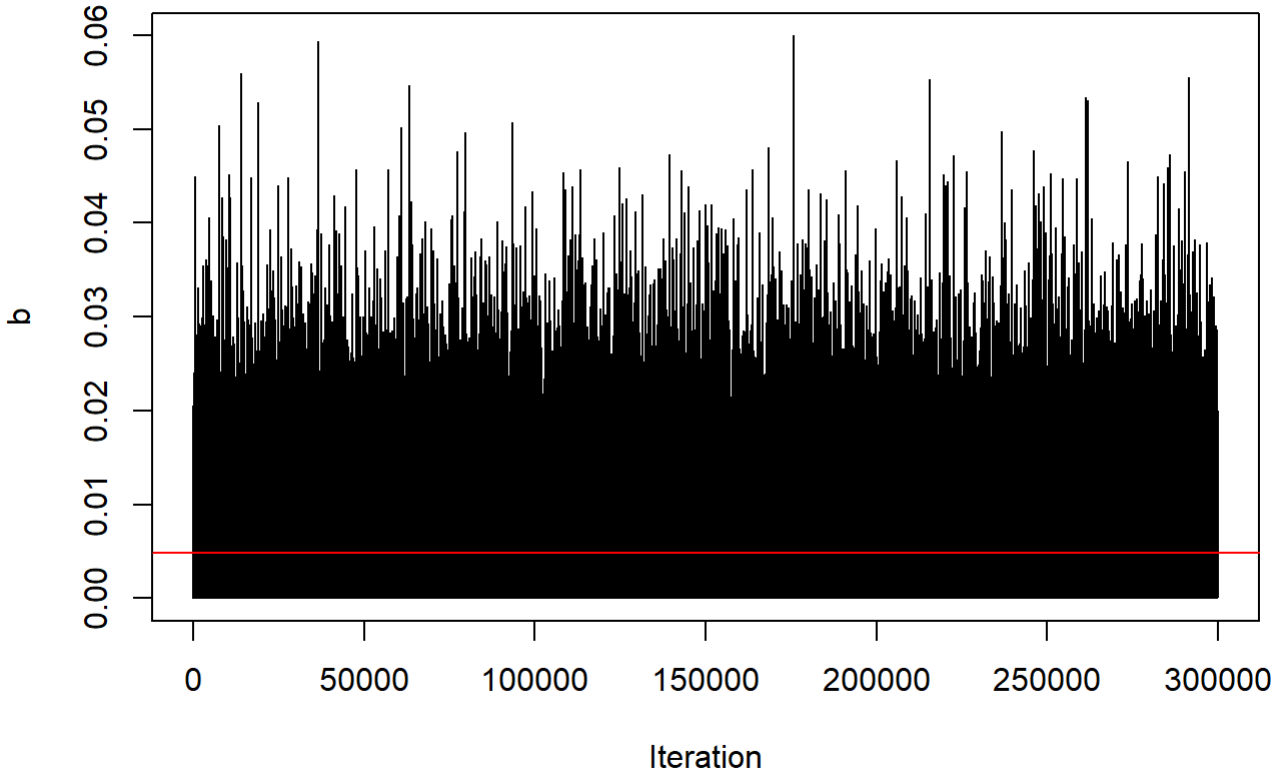
```
## ##
## ## Support provided by the U.S. National Science Foundation
```

```
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
# Updating sigma2
sigma2.update <- function(y,a,b)
{
  out <- rinvgamma(n = 10,shape = a + 0.5,scale = 0.5*y^2 + b)
  return(out)
}

# updating b, but posterior of b is independent of sigma2 and y
b.update <- function(y,sigma2)
{
  out <- rexp(1,rate = (sum(1/sigma2)) + 1)
  return(out)
}

MCMC <- function(y,a,sigma2.init,b.init,iters)
{
  #chain initiation
  sigma2 <- sigma2.init
  b <- b.init

  # define chains
  sigma2.chain <- matrix(NA,nrow=iters,ncol=10)
  b.chain <- rep(NA,iters)

  # start MCMC
  for(i in 1:iters)
  {
    sigma2 <- sigma2.update(y,a,b)
    b <- b.update(y,sigma2)

    sigma2.chain[i,] <- sigma2
    b.chain[i] <- b
  }

  # return chains
  out <- list(sigma2.chain = sigma2.chain, b.chain = b.chain)
  return(out)
}

y <- 1:10
MCMC.out <- MCMC(y = y[1],a = 10,sigma2.init = rep(var(y),10),b.init = 0,iters = 3e5)

plot(MCMC.out$sigma2.chain[,1],xlab = "Iteration",ylab = "sigma2",type = "l")
abline(h = mean(MCMC.out$sigma2.chain),col = "red")
```



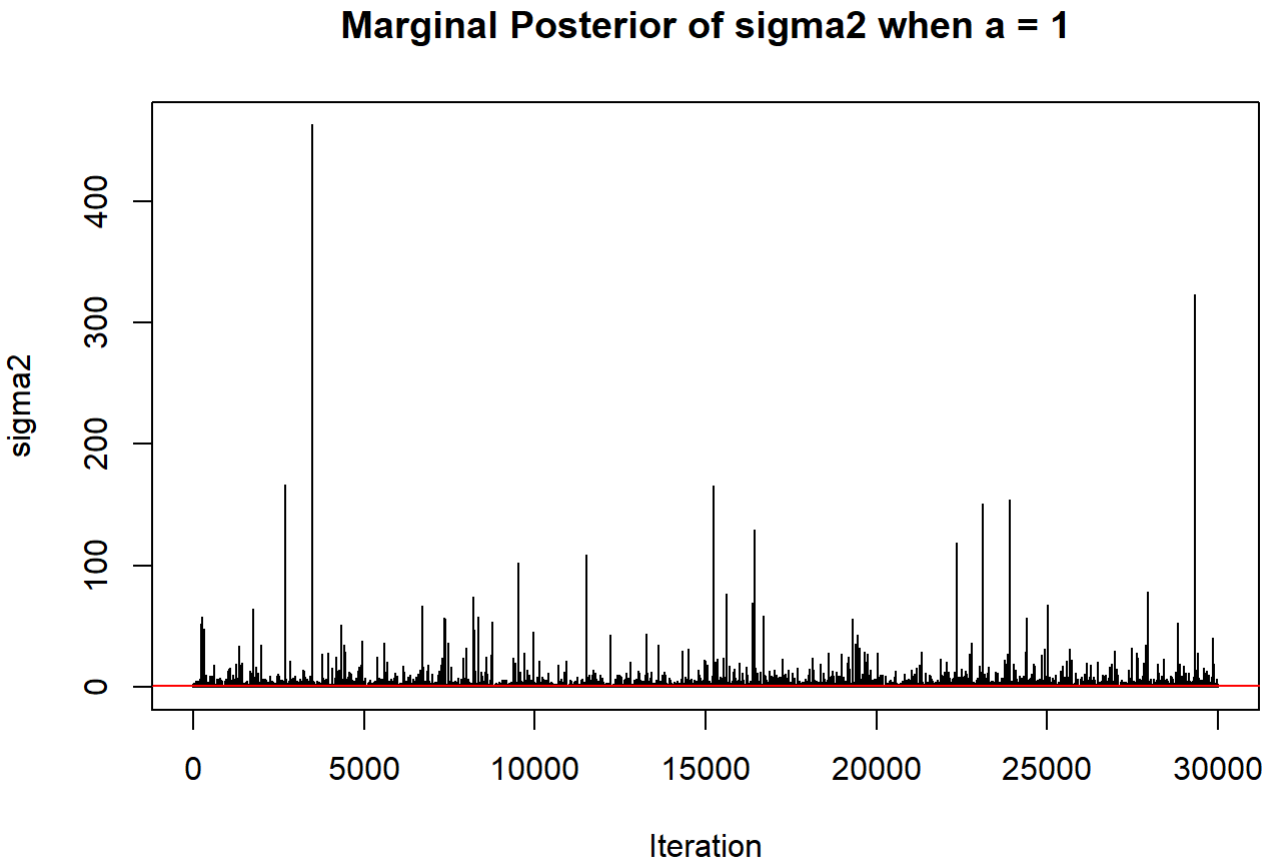

```
plot(MCMC.out$b.chain,xlab = "Iteration",ylab = "b",type = "l")
abline(h = mean(MCMC.out$b.chain),col = "red")
```



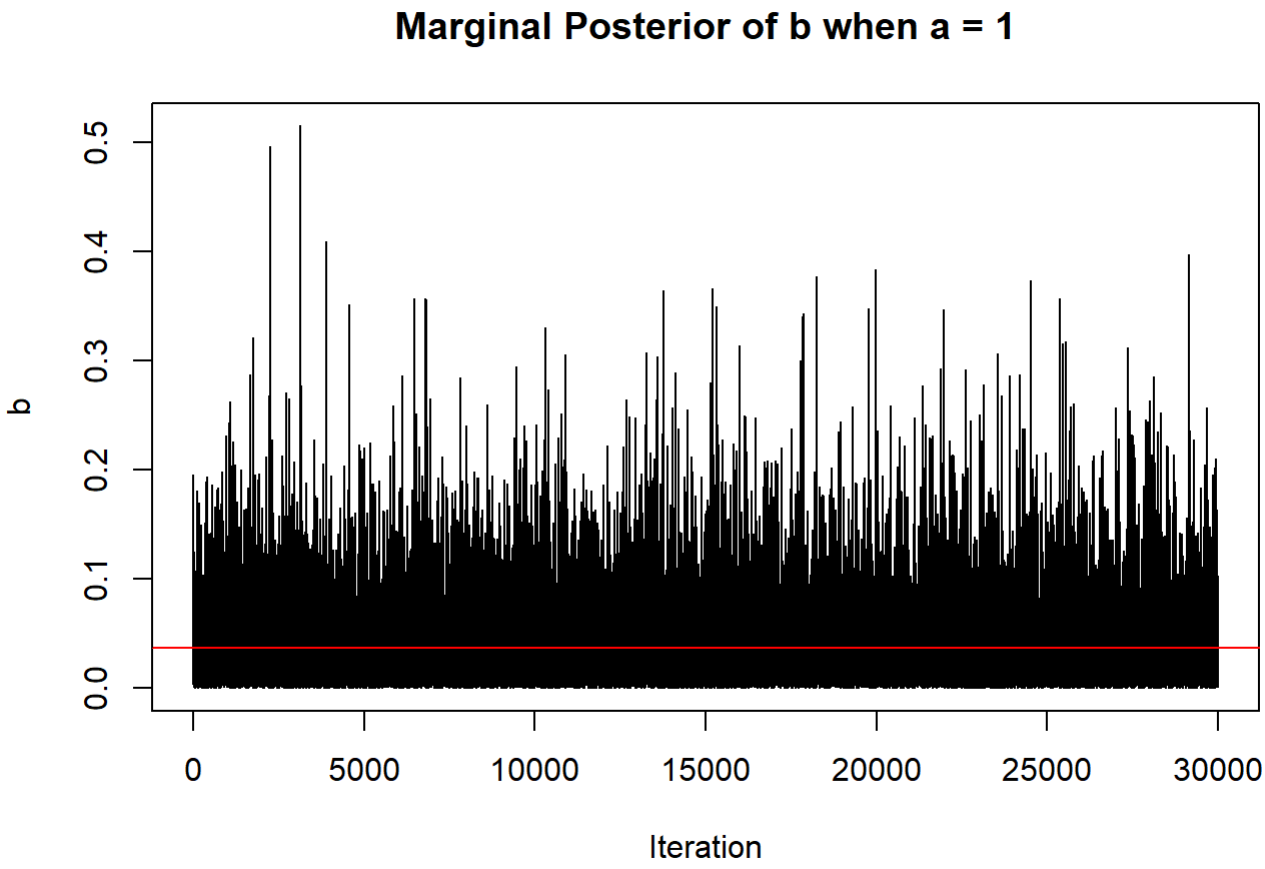
(d)

For $a = 1$

```
set.seed(123)
y <- 1:10
MCMC.out <- MCMC(y = y[1],a = 1,sigma2.init = rep(var(y),10),b.init = 0,iters = 3e4)

plot(MCMC.out$sigma2.chain[,1],xlab = "Iteration",ylab = "sigma2",type = "l",main ="Marginal
Posterior of sigma2 when a = 1")
abline(h = mean(MCMC.out$sigma2.chain),col = "red")
```




```
plot(MCMC.out$b.chain,xlab = "Iteration",ylab = "b",type = "l",main ="Marginal Posterior of b
when a = 1")
abline(h = mean(MCMC.out$b.chain),col = "red")
```



- Marginal Posterior of σ_1^2:
  - As a changes from 10 to 1, the shape parameter of posterior distribution of σ_1^2 decreases, the samples of σ_1^2 increases.
- Marginal Posterior of b:
  - Changing the value of a from 10 to 1 does not directly affect the update of b.
  - As σ_1^2 changes due to change in a, the samples of b increases.

(e)

In JAGS,

Question - 4

(a)

- **Option 1 (Emphasizing Player Variability)**
  - This Beta distribution prior for θ_1 effectively encapsulates the wide spectrum of shooting performances among players. Acting as a conjugate prior to the Binomial likelihood, it offers a versatile framework for modeling success probabilities, making it an ideal choice to represent the diverse skill levels of players in clutch situations.
- **Option 2 (Accentuating Conjugate Prior's Benefit)**
  - An essential feature of this model stems from the Beta distribution selected as the prior for θ_1. Serving as a conjugate prior to the Binomial likelihood, the Beta distribution provides flexibility in modeling success probabilities. This adaptability is pivotal in encompassing the varied shooting abilities observed among NBA players during crucial free throw attempts.

(b)

The parameter m within the prior governs both the central tendency and dispersion of the Beta distribution. By applying an exponential transformation to m, we guarantee that the resulting shape parameters for the Beta distribution $(exp(m) * q_i$ and $exp(m) * (1 - q_i))$ remain positive, preserving the integrity of the Beta distribution. Introducing a Normal distribution for m with a mean of 0 and a standard deviation of 10 enables the representation of variability in overall shooting performance among players, where higher m values signify greater overall success rates and vice versa. Thus, m holds a pivotal role in shaping the prior distribution and shaping prior beliefs regarding players' success rates in clutch shots.

(c)

Full conditional distribution of θ_1 is Beta$(y_1 + e^m q_1, n_1 - y_1 + e^m(1 - q_1))$

(d)

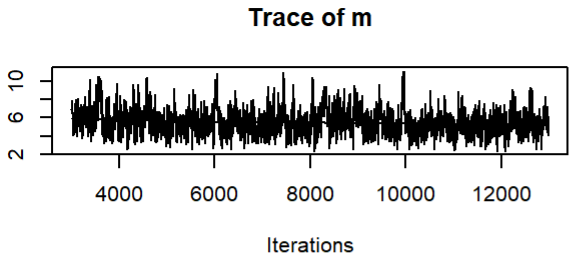
(e)

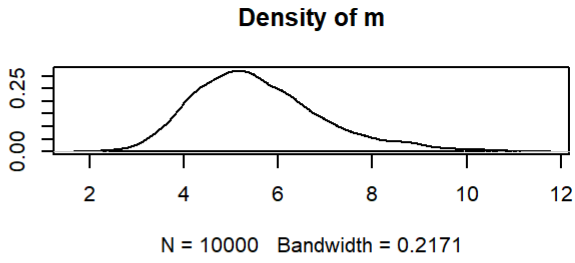
Using JAGS

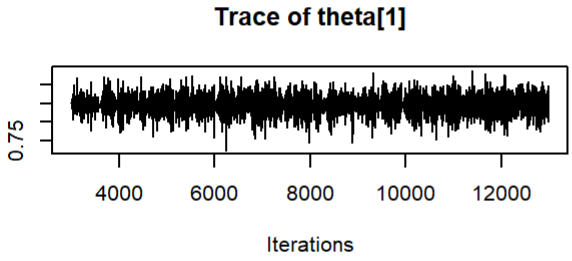
```
## Linked to JAGS 4.3.1
```

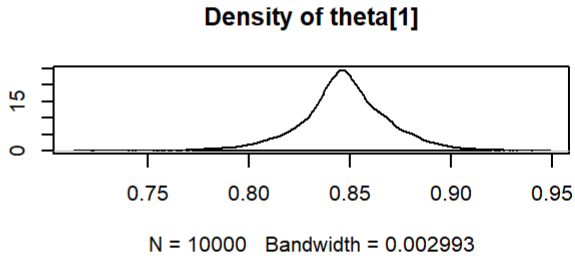
```
## Loaded modules: basemod,bugs
```

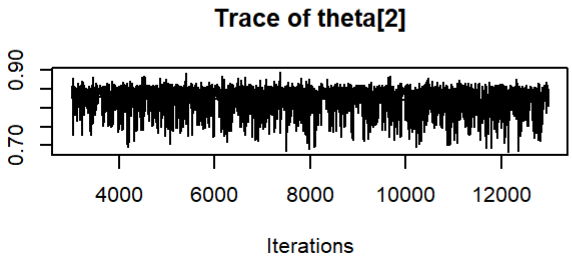
```
##
## Iterations = 3001:13000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean      SD  Naive SE Time-series SE
## m         5.5708 1.36471 0.0136471      0.0792077
## theta[1]  0.8473 0.02218 0.0002218      0.0003075
## theta[2]  0.8184 0.02946 0.0002946      0.0009516
## theta[3]  0.8784 0.02113 0.0002113      0.0003282
## theta[4]  0.6775 0.03278 0.0003278      0.0004834
## theta[5]  0.9075 0.01778 0.0001778      0.0002765
## theta[6]  0.9018 0.02162 0.0002162      0.0003656
## theta[7]  0.7531 0.03231 0.0003231      0.0007593
## theta[8]  0.8021 0.02404 0.0002404      0.0003326
## theta[9]  0.7874 0.03020 0.0003020      0.0006164
## theta[10] 0.8696 0.02655 0.0002655      0.0004363
##
## 2. Quantiles for each variable:
##
##             2.5%    25%    50%    75%  97.5%
## m         3.3966 4.6016 5.3902 6.3331 8.8307
## theta[1]  0.7994 0.8359 0.8472 0.8598 0.8922
## theta[2]  0.7473 0.8020 0.8246 0.8410 0.8585
## theta[3]  0.8300 0.8678 0.8796 0.8905 0.9184
## theta[4]  0.6081 0.6611 0.6762 0.6948 0.7470
## theta[5]  0.8664 0.8983 0.9087 0.9178 0.9421
## theta[6]  0.8560 0.8909 0.9008 0.9136 0.9465
## theta[7]  0.6751 0.7376 0.7593 0.7735 0.8039
## theta[8]  0.7501 0.7892 0.8022 0.8159 0.8498
## theta[9]  0.7110 0.7734 0.7933 0.8060 0.8355
## theta[10] 0.8049 0.8582 0.8728 0.8838 0.9175
```

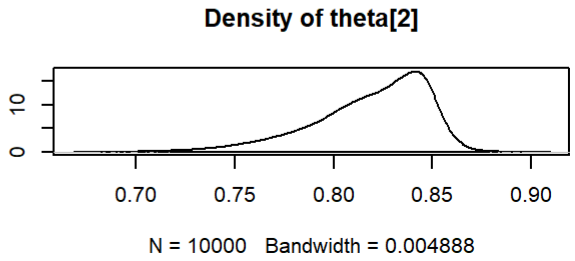

Trace of m
2
6
10
4000
6000
8000
10000
12000
Iterations

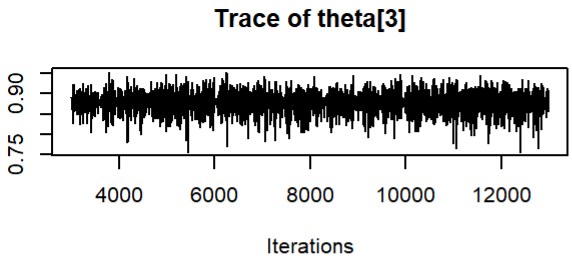

Density of m
0.00
0.25
2
4
6
8
10
12
N = 10000 Bandwidth = 0.2171

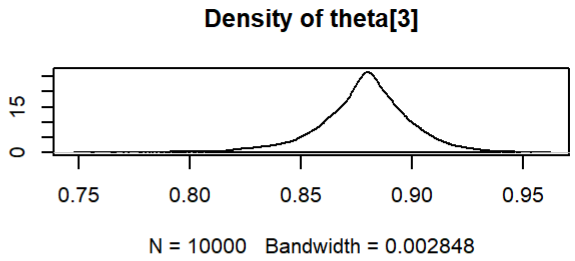

Trace of theta[1]
0.75
4000
6000
8000
10000
12000
Iterations

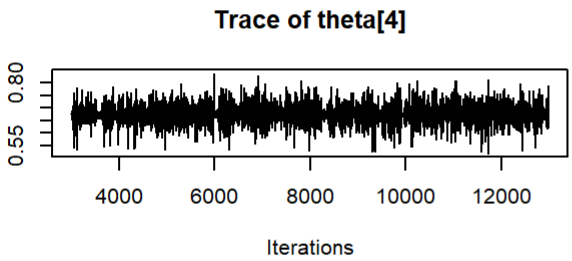

Density of theta[1]
0
15
0.75
0.80
0.85
0.90
0.95
N = 10000 Bandwidth = 0.002993

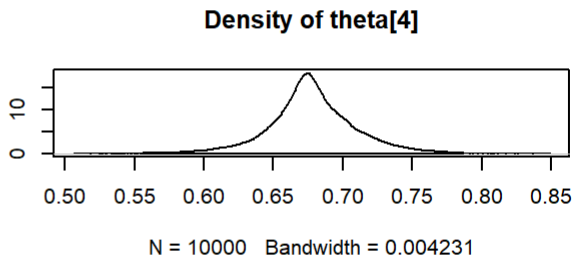

Trace of theta[2]
0.70
0.90
4000
6000
8000
10000
12000
Iterations

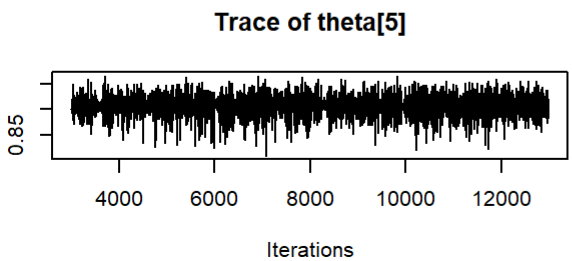

Density of theta[2]
0
10
0.70
0.75
0.80
0.85
0.90
N = 10000 Bandwidth = 0.004888

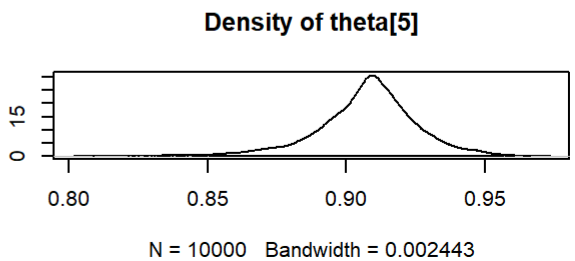

Trace of theta[3]
0.75
0.90
4000
6000
8000
10000
12000
Iterations

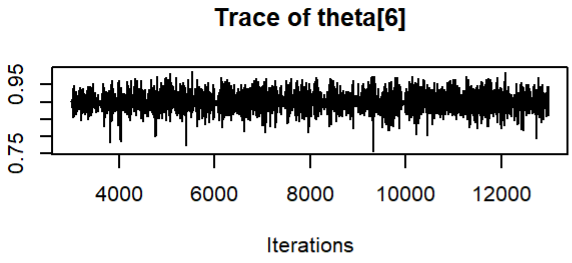

Density of theta[3]
0
15
0.75
0.80
0.85
0.90
0.95
N = 10000 Bandwidth = 0.002848

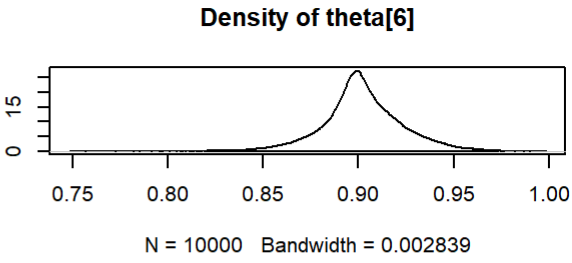

Trace of theta[4]
0.55
0.80
4000
6000
8000
10000
12000
Iterations

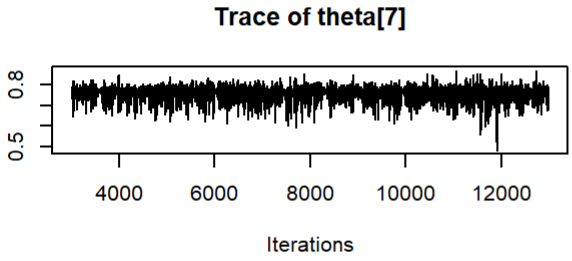

Density of theta[4]
0
10
0.50
0.55
0.60
0.65
0.70
0.75
0.80
0.85
N = 10000 Bandwidth = 0.004231

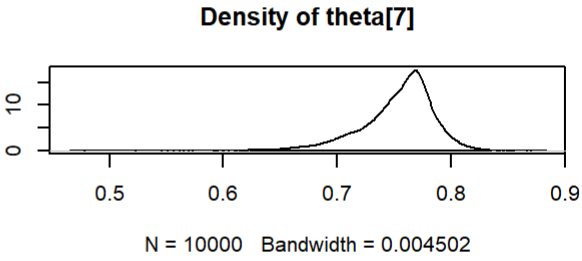

Trace of theta[5]
0.85
4000
6000
8000
10000
12000
Iterations

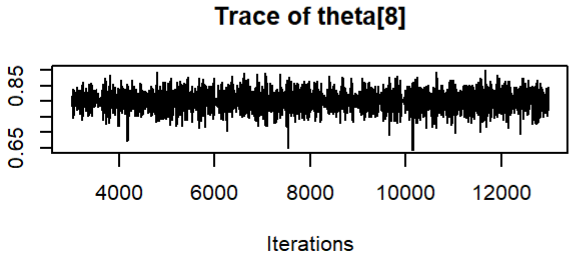

Density of theta[5]
0
15
0.80
0.85
0.90
0.95
N = 10000 Bandwidth = 0.002443

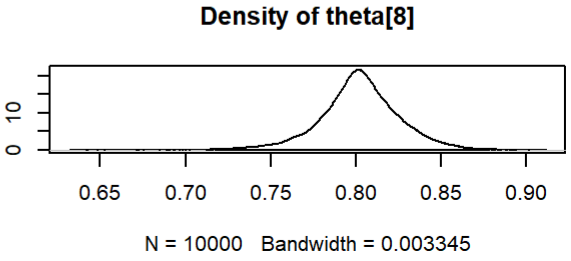


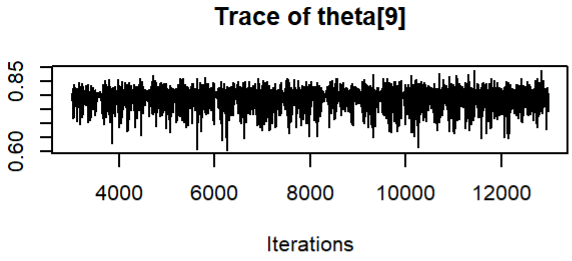


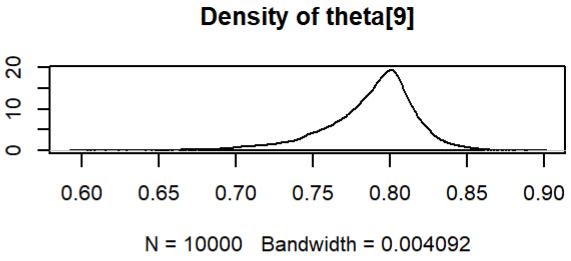


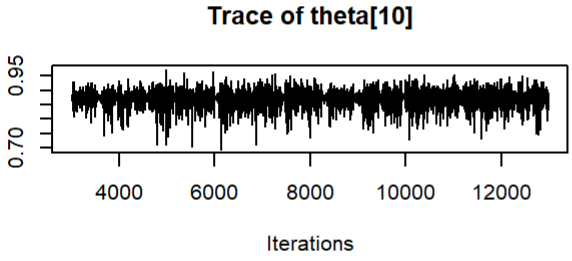


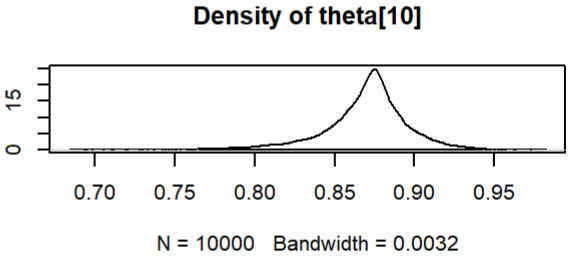












(f)

Custom MCMC Sampling:

- Advantages:
    - Flexibility: With custom code, you have full control over the model specification and sampling procedure. You can easily customize the algorithm to fit specific requirements or experimental designs.

```
- Learning Experience: Implementing MCMC algorithms from scratch helps in understanding the u
nderlying concepts of Bayesian inference and MCMC methods. It provides a deeper insight into
how the algorithms work.

-  Efficiency for Simple Models: For relatively simple models and small datasets, custom MCMC
code can be efficient and straightforward to implement. It avoids the overhead of setting up
and running external software like JAGS or Stan.
```

- Disadvantages:
    - Complexity for Complex Models: Writing custom MCMC code becomes increasingly complex for more intricate models with high-dimensional parameter spaces or non-standard likelihood/prior distributions. Debugging and optimizing such code can be time-consuming.
    - Verification and Validation: Custom MCMC code requires rigorous testing, verification, and validation to ensure correctness and reliability. Without thorough testing, there's a risk of errors leading to incorrect inferences.

JAGS (Just Another Gibbs Sampler):

- Advantages:
    - Ease of Use: JAGS provides a high-level modeling language (BUGS syntax) that simplifies model specification. It offers an intuitive and user-friendly interface for defining Bayesian models, making it accessible to researchers without extensive programming experience.
    - Efficiency for Complex Models: JAGS is well-suited for complex Bayesian models with intricate structures, hierarchical dependencies, and large datasets. It efficiently handles sophisticated models without requiring manual tuning of sampling algorithms.
    - Validation and Community Support: JAGS has been extensively validated and tested, providing confidence in its correctness and reliability. It benefits from a large user community, extensive documentation, and online support resources.
- Disadvantages:
    - Limited Flexibility: While JAGS provides a wide range of modeling capabilities, it may not support certain advanced features or custom sampling algorithms required for highly specialized models.
    - External Dependency: JAGS is an external software package that needs to be installed separately. It introduces a dependency on external software, potentially requiring additional setup and management.

Question - 5

(a)

Given $Y_i|$ Laplace(μ, σ) for $i = 1, 2, . . , n$ where $\theta = (\mu, \sigma)$

Probability density function of Laplace is

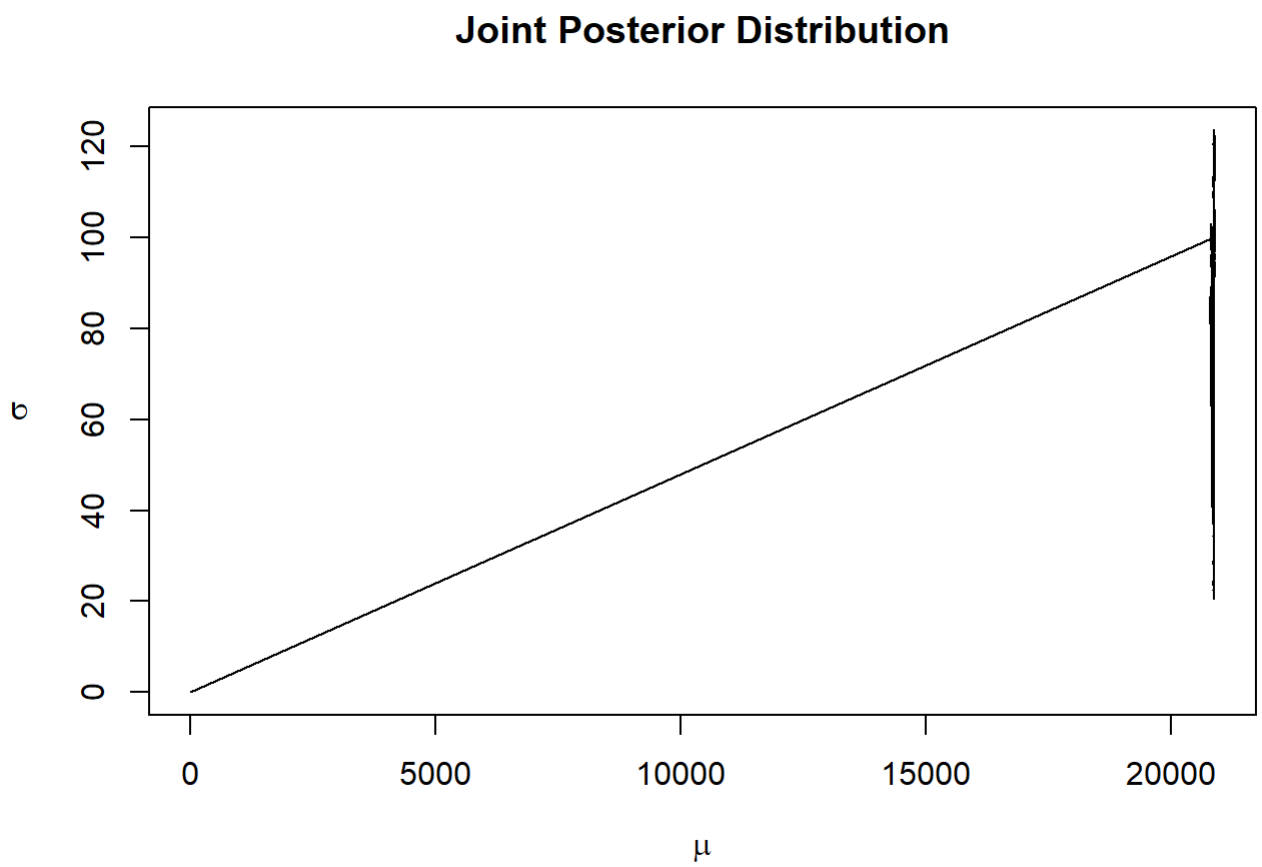
$$f(x) = \frac{1}{2\sigma}\exp(-\frac{|x-\mu|}{\sigma})$$

Given improper prior

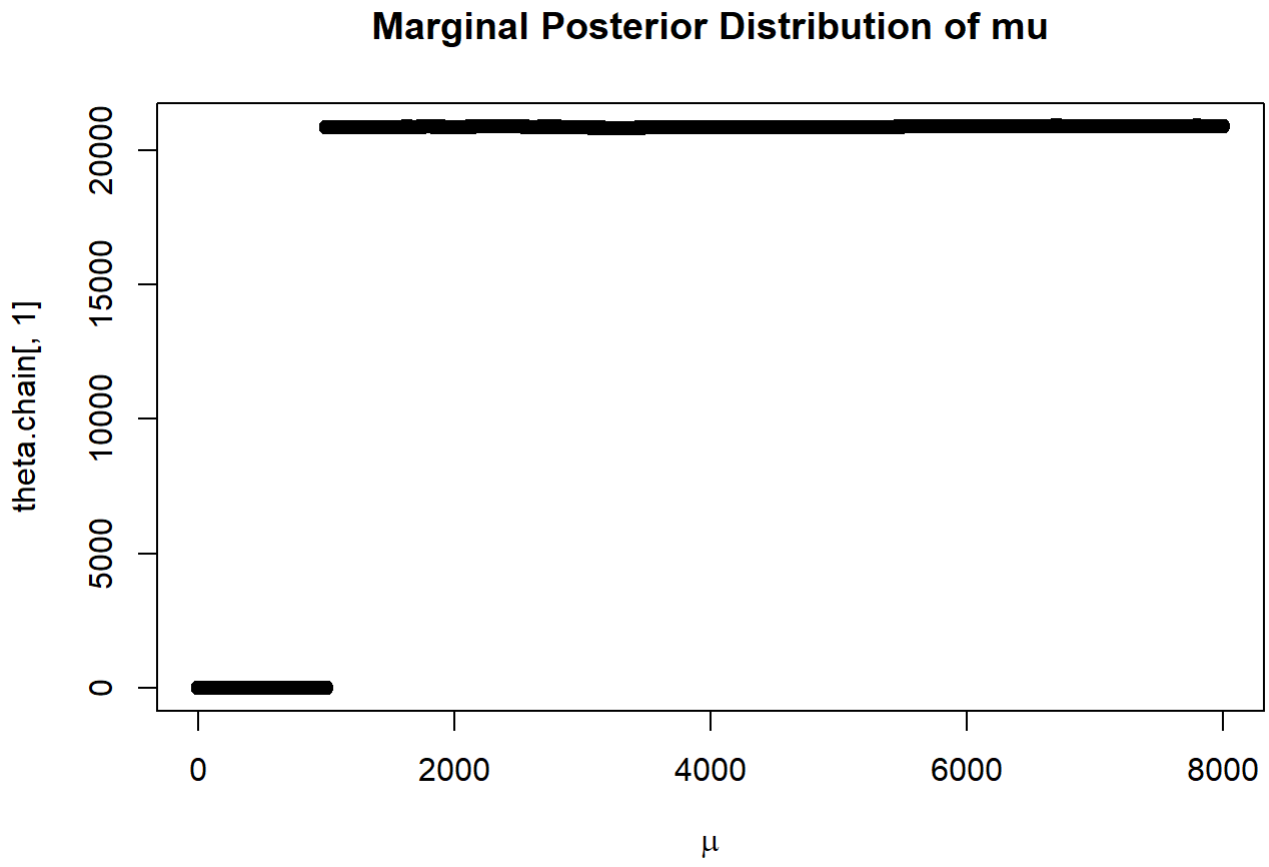
- $\sigma \sim$ Uniform(0,100000)
- $\pi(\mu) = 1$ for all $\mu \in (-\infty, \infty)$

```
set.seed(123)
library(MASS)
data(galaxies)
Y <- galaxies

# likelihood function
likelihood <- function(mu, sigma, y) {
  n <- length(y)
  log_likelihood <- -n * log(2 * sigma) - sum(abs(y - mu) / sigma)
  return(exp(log_likelihood))
}

# prior for mu
prior_mu <- function(mu){
  return (1)
}

# prior for sigma
prior_sigma <- function(sigma) {
  ifelse(sigma <= 0 | sigma > 100000,0,1)
}

# posterior distribution
posterior <- function(mu, sigma, y) {
  return(likelihood(mu, sigma, y) * prior_sigma(sigma)*prior_mu(mu))
}

# MCMC sampling using Metropolis-Hastings algorithm
n_samples <- 8000
burn_in <- 1000

theta.init <- c(mean(Y), 100)  # Initial guess for theta = (mu,sigma)

# Metropolis-Hastings loop
theta.chain <- matrix(0, n_samples, 2)
accept.count <- 0
for (i in (burn_in + 1):n_samples) {
  # Propose new candidate
  mu <-  rnorm(1, mean = theta.init[1], 0.8)
  sigma <-  rnorm(1, mean = theta.init[2], 0.8)
  theta.prop <- c(mu, sigma)

  # Calculate acceptance probability
  R <- exp(posterior(mu, sigma, Y) - posterior(theta.init[1],theta.init[1],Y))
  u <- runif(1)

  if (u < R) {
    theta.init <- theta.prop
    accept.count <- accept.count + 1
  }

  # Store sample
  theta.chain[i, ] <- theta.init
}
```

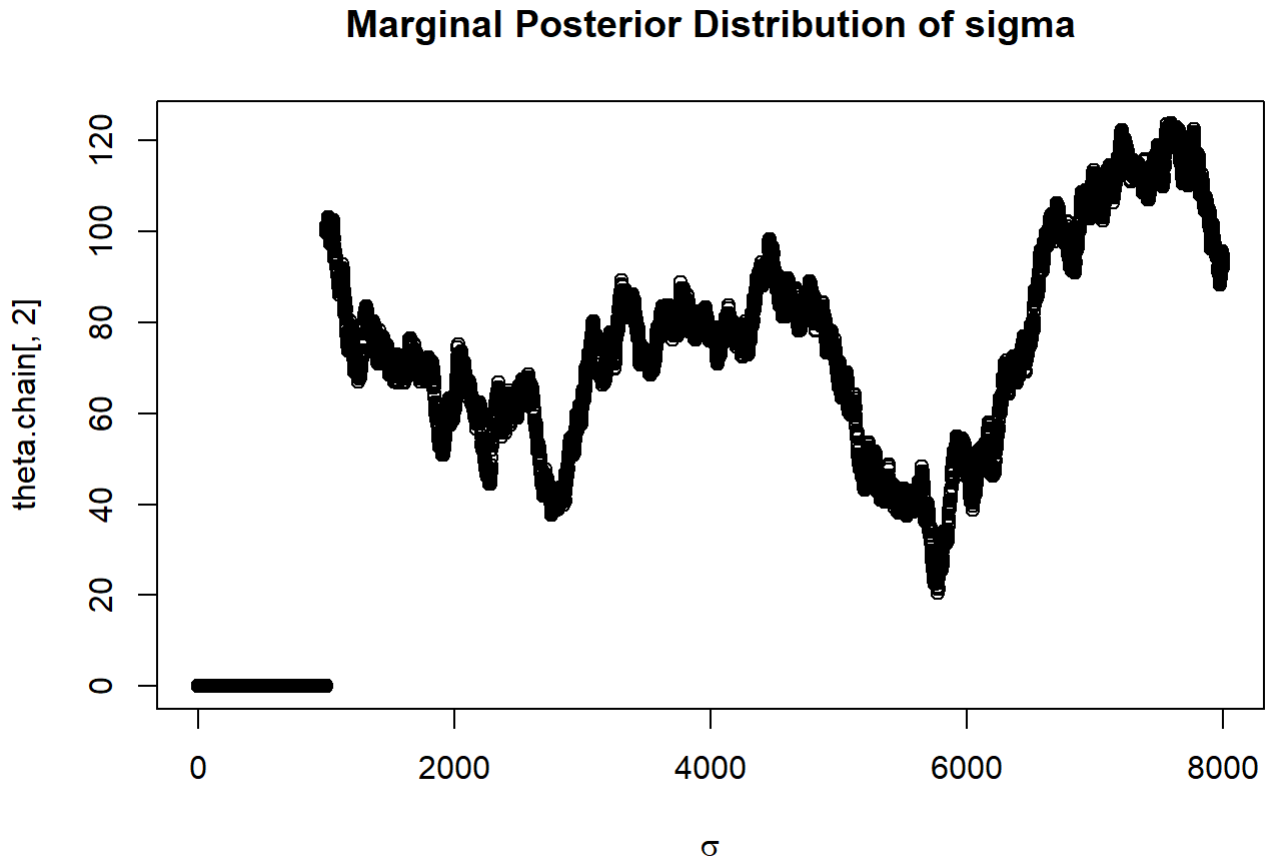
```
plot(theta.chain[,1], theta.chain[,2], xlab=expression(mu), ylab=expression(sigma), main="Joi
nt Posterior Distribution",type="l")
```




```
# Plot marginal posterior distributions
plot(theta.chain[,1], col="black", xlab=expression(mu), main="Marginal Posterior Distribution
of mu")
```



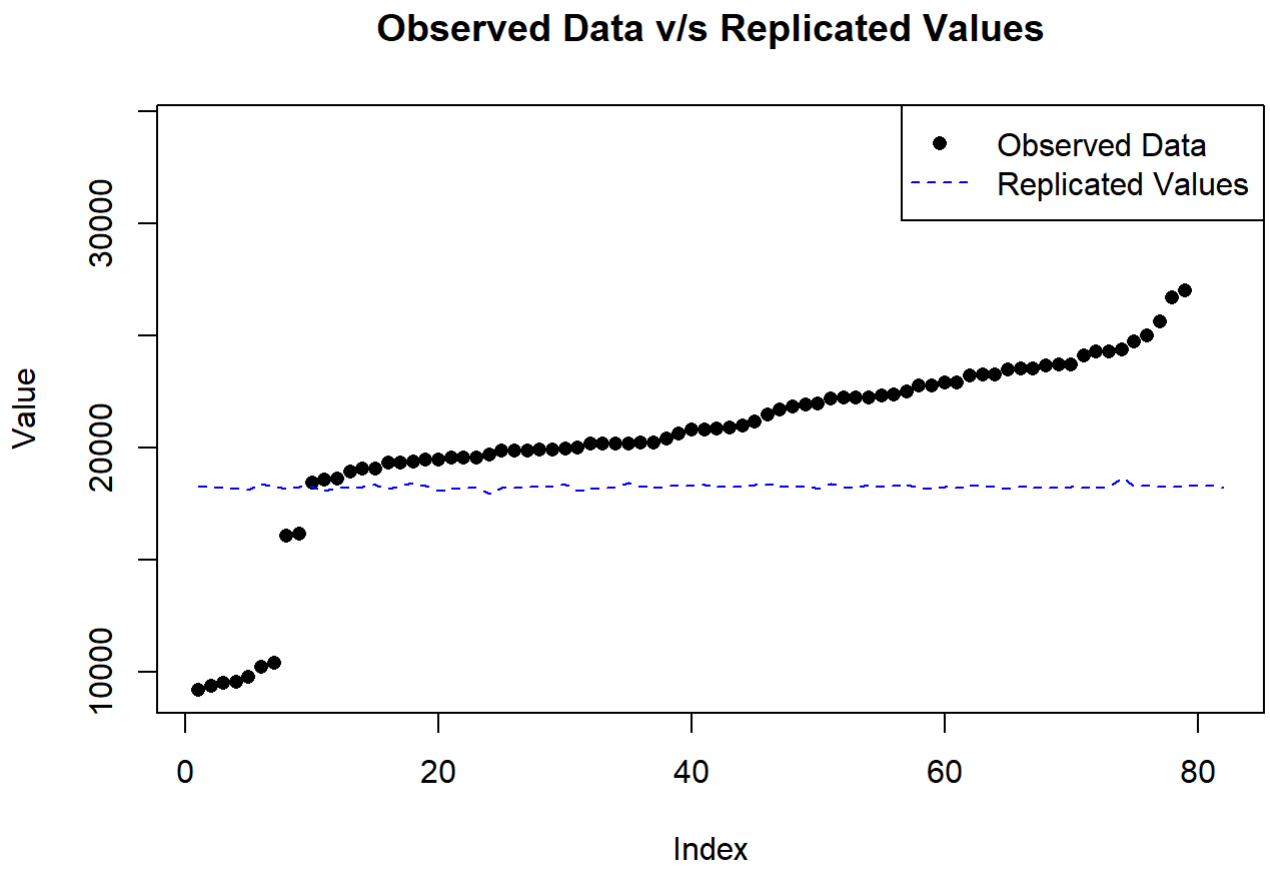

```
plot(theta.chain[,2], col="black", xlab=expression(sigma), main="Marginal Posterior Distribut
ion of sigma")
```



(b)

Posterior mean of θ and plot Laplace PDF values against the observed data.

```
set.seed(123)
# Calculate posterior mean of theta
mu_mean <- mean(theta.chain[,1])
sigma_mean <- mean(theta.chain[,2])
theta_mean <- c(mu_mean, sigma_mean)

# Function to generate Laplace distributed samples
rLaplace <- function(n, mu, sigma) {
  u <- runif(n, 0, 1)
  ifelse(u <= 0.5, mu - sigma * log(2 * u), mu + sigma * log(2 * (1 - u)))
}

# Generate data points from Laplace distribution with posterior mean parameters
y_replicated <- rLaplace(length(Y), mu_mean, sigma_mean)


# Plot data and replicated values
plot(Y, type = "p", col = "black", pch = 16, ylim = range(c(Y, y_replicated)),
     xlab = "Index", ylab = "Value", main = "Observed Data v/s Replicated Values")
lines(y_replicated, col = "blue", lty = 2)
legend("topright", legend = c("Observed Data", "Replicated Values"),
       col = c("black", "blue"), pch = c(16, NA), lty = c(NA, 2))
```

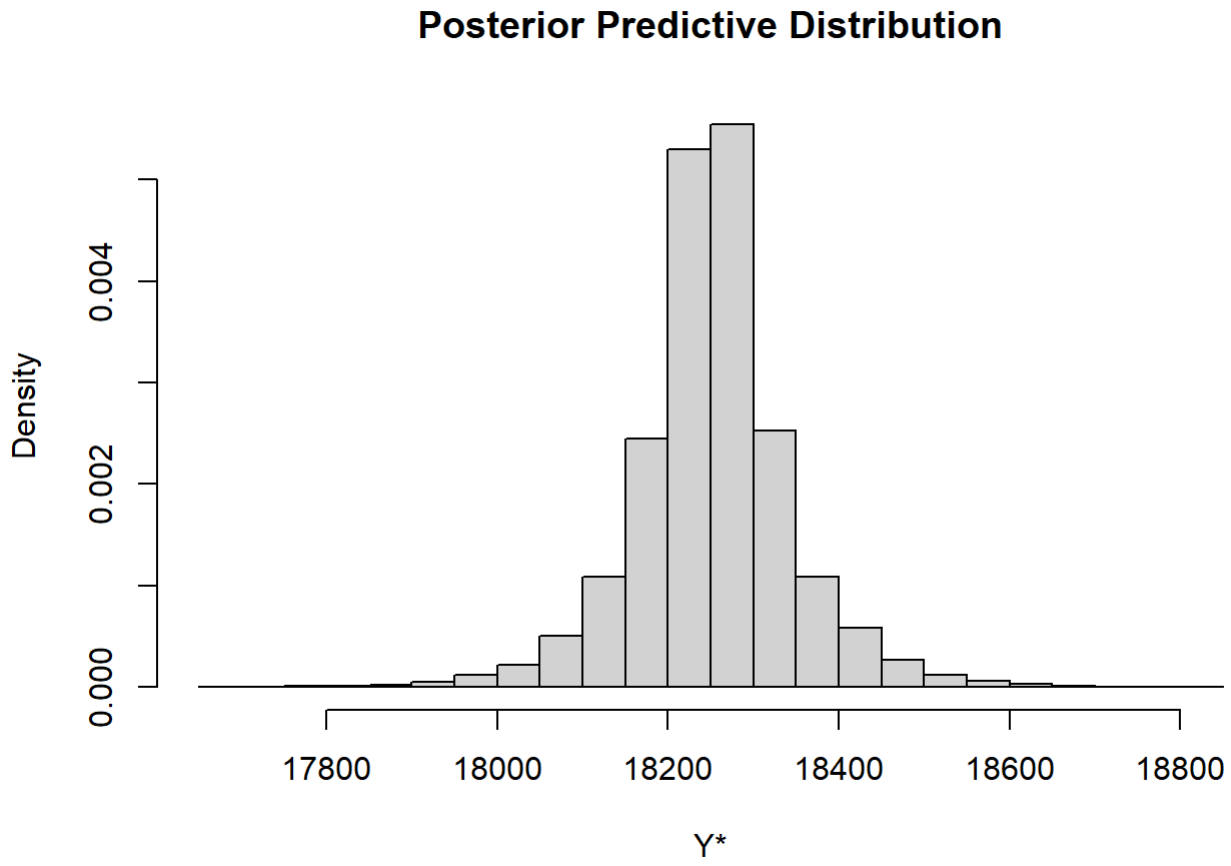


No, the model donot fit the data well.

(c)

posterior predictive distribution (PPD) for a new observation $Y^{*}|$ Laplace(μ, σ).

```
set.seed(123)
# new observations
y_new <- rLaplace(10000, mu_mean, sigma_mean)

# Plot ppd
hist(y_new, breaks = 30, prob = TRUE,
     main = "Posterior Predictive Distribution", xlab = "Y*")
```




```
ppd_mean <- mean(y_new)
ppd_variance <- var(y_new)

plugin_mean <- mu_mean
plugin_variance <- 2 * sigma_mean^2  # Variance of Laplace distribution = 2 * sigma^2

distribution_data <- data.frame(
  Distribution = c("Posterior Predictive Distribution", "Plug-in Distribution"),
  Mean = c(ppd_mean, plugin_mean),
  Variance = c(ppd_variance, plugin_variance)
)

distribution_data
```

```
##                        Distribution     Mean Variance
## 1 Posterior Predictive Distribution 18251.66 8451.311
## 2              Plug-in Distribution 18250.73 8603.052
```

Mean and variance of corresponding ppd and plug-in are mostly similar