Assignment - 5

Soni Verma

2024-04-15

Question - 1

a part

```
library(geoR)
```

```
## ----------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.9-4 (built on 2024-02-14) is now loaded
## ----------------------------------------------------------
```

```
data("gambia")

Y = gambia$pos
X = gambia[,-3]
n = length(Y)
p = ncol(X)

library(rjags)
```

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

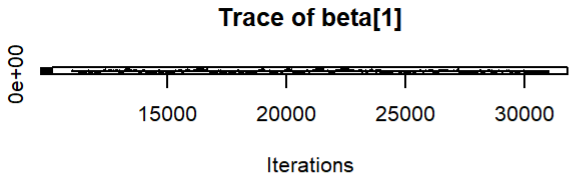
```
## Linked to JAGS 4.3.1
```

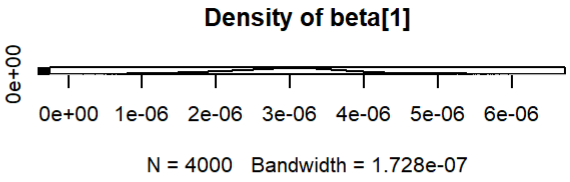
```
## Loaded modules: basemod,bugs
```

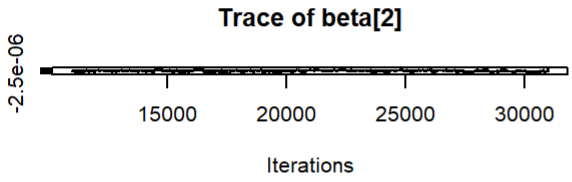
```
#part a
data = list(Y = Y, n = n, p = p, X = X)
params = c('beta')

model_string_1 = textConnection("model{
      #Likelihood
      for(i in 1:n){
      Y[i] ~ dbinom(probs[i], 1)
      logit(probs[i]) = inprod(X[i,], beta)
      }
      #Priors
      for(j in 1:p){
      beta[j] ~ dnorm(0, 1e-3)
      }
}")

model_1 = jags.model(model_string_1, data = data,
                     quiet = TRUE)
update(model_1, 1e4)
samples1 = coda.samples(model_1, variable.names = params,
                        thin = 5, n.iter = 2e4)
summary(samples1)
```

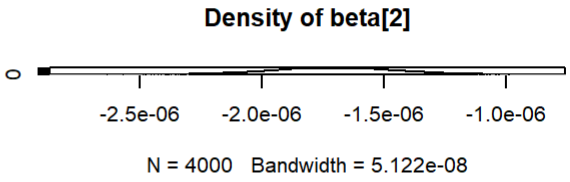
```
##
## Iterations = 11005:31000
## Thinning interval = 5
## Number of chains = 1
## Sample size per chain = 4000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean        SD  Naive SE Time-series SE
## beta[1]  2.998e-06 9.479e-07 1.499e-08      1.070e-07
## beta[2] -1.706e-06 2.538e-07 4.013e-09      2.244e-08
## beta[3]  6.576e-04 1.120e-04 1.771e-06      3.477e-06
## beta[4] -5.626e-01 1.104e-01 1.745e-03      3.700e-03
## beta[5] -2.294e-01 1.337e-01 2.113e-03      3.360e-03
## beta[6]  8.101e-03 1.337e-02 2.113e-04      2.075e-03
## beta[7] -3.013e-01 1.161e-01 1.836e-03      3.804e-03
##
## 2. Quantiles for each variable:
##
##               2.5%        25%        50%        75%      97.5%
## beta[1]  1.085e-06  2.403e-06  3.006e-06  3.550e-06  5.066e-06
## beta[2] -2.207e-06 -1.877e-06 -1.707e-06 -1.534e-06 -1.205e-06
## beta[3]  4.394e-04  5.817e-04  6.569e-04  7.333e-04  8.765e-04
## beta[4] -7.813e-01 -6.350e-01 -5.624e-01 -4.910e-01 -3.494e-01
## beta[5] -4.984e-01 -3.168e-01 -2.276e-01 -1.384e-01  1.745e-02
## beta[6] -2.127e-02  3.328e-04  7.708e-03  1.655e-02  3.586e-02
## beta[7] -5.277e-01 -3.797e-01 -3.027e-01 -2.194e-01 -8.054e-02
```

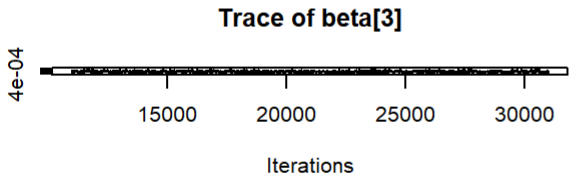
```
plot(samples1)
```

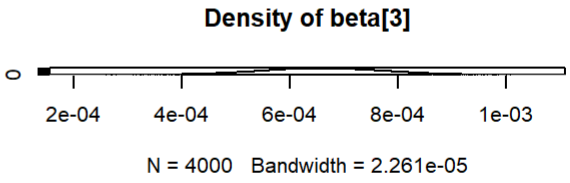

Trace of beta[1]
0e+00
15000
20000
25000
30000
Iterations

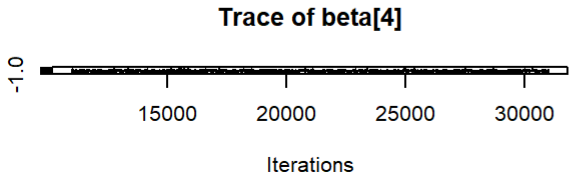

Density of beta[1]
0e+00
0e+00
1e-06
2e-06
3e-06
4e-06
5e-06
6e-06
N = 4000 Bandwidth = 1.728e-07

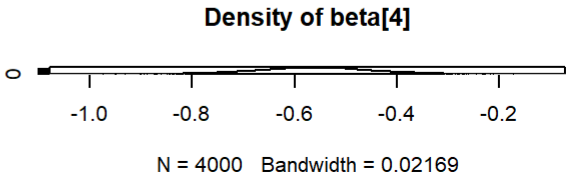

Trace of beta[2]
-2.5e-06
15000
20000
25000
30000
Iterations

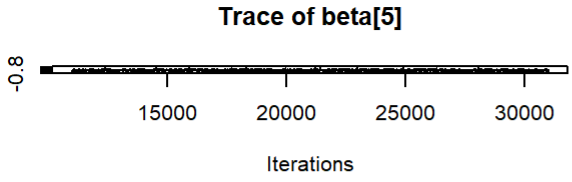

Density of beta[2]
0
-2.5e-06
-2.0e-06
-1.5e-06
-1.0e-06
N = 4000 Bandwidth = 5.122e-08

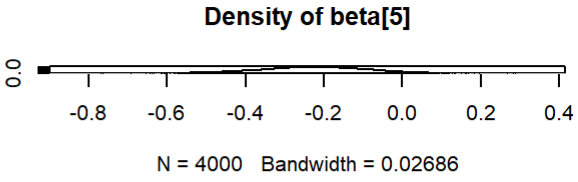

Trace of beta[3]
4e-04
15000
20000
25000
30000
Iterations

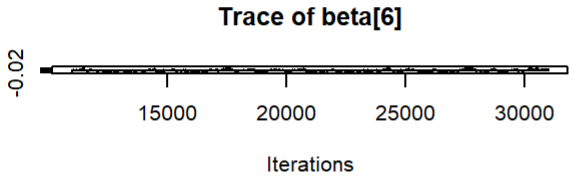

Density of beta[3]
0
2e-04
4e-04
6e-04
8e-04
1e-03
N = 4000 Bandwidth = 2.261e-05

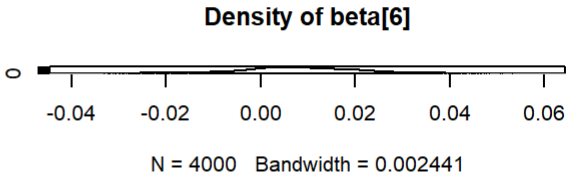

Trace of beta[4]
-1.0
15000
20000
25000
30000
Iterations

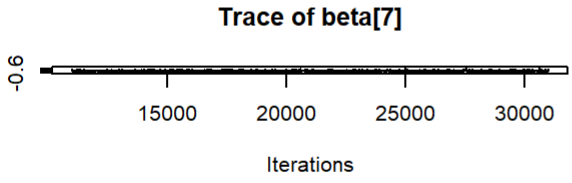

Density of beta[4]
0
-1.0
-0.8
-0.6
-0.4
-0.2
N = 4000 Bandwidth = 0.02169

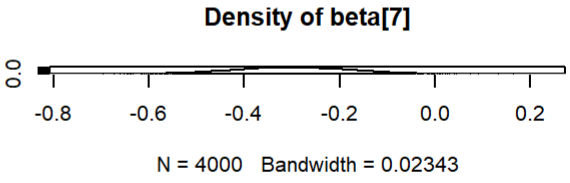

Trace of beta[5]
-0.8
15000
20000
25000
30000
Iterations


Density of beta[5]
0.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
N = 4000 Bandwidth = 0.02686


Trace of beta[6]
-0.02
15000
20000
25000
30000
Iterations

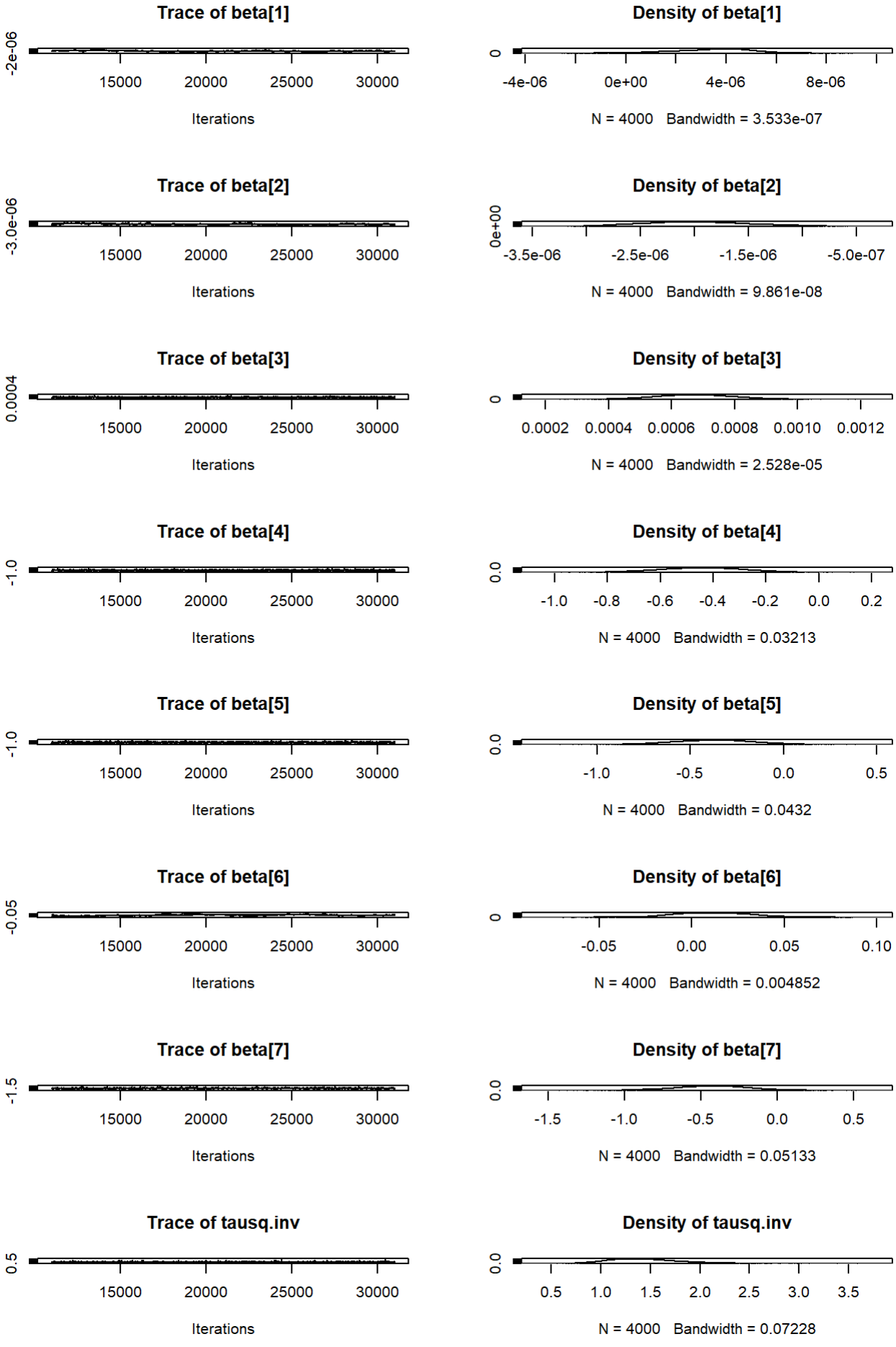

Density of beta[6]
0
-0.04
-0.02
0.00
0.02
0.04
0.06
N = 4000 Bandwidth = 0.002441


Trace of beta[7]
-0.6
15000
20000
25000
30000
Iterations


Density of beta[7]
0.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
N = 4000 Bandwidth = 0.02343

b part

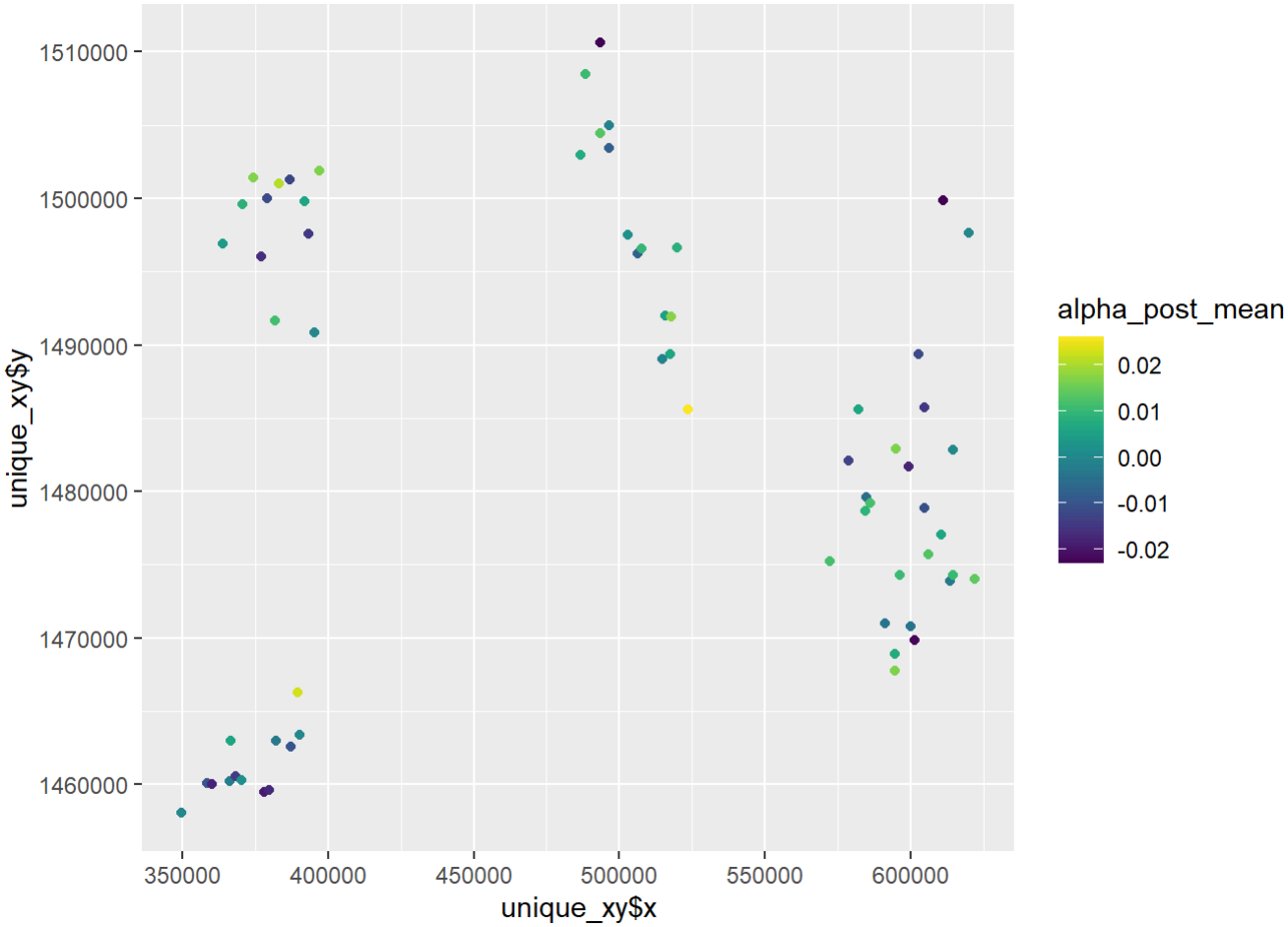
```
library(tidyr)
params = c('beta', 'tausq.inv')
unique_xy = unique(gambia[,1:2])
s = nrow(unique_xy)
index = 1 : s
s_ind =  index[match(paste(gambia$x, gambia$y), paste(unique_xy$x, unique_xy$y))]

data = list(Y = Y, n = n, p = p, X = X, s_ind = s_ind, s = s)

model_string_2 = textConnection("model{
      #Likelihood
      for(i in 1:n){
      Y[i] ~ dbinom(probs[i], 1)
      logit(probs[i]) = inprod(X[i,], beta) + alpha[s_ind[i]]
      }

      #Random_Effects
      for(i in 1 : s){
      alpha[i] ~ dnorm(0, tausq.inv)
      }

      #Priors
      for(j in 1:p){
      beta[j] ~ dnorm(0, 1e-3)
      }
      tausq.inv ~ dgamma(0.01, 0.01)
}")

model_2 = jags.model(model_string_2, data = data,
                     quiet = TRUE)
update(model_2, 1e4)
samples2 = coda.samples(model_2, variable.names = params,
                        thin = 5, n.iter = 2e4)
plot(samples2)
```


Trace of beta[1]
-2e-06
15000
20000
25000
30000
Iterations
Density of beta[1]
0
-4e-06
0e+00
4e-06
8e-06
N = 4000 Bandwidth = 3.533e-07
Trace of beta[2]
-3.0e-06
15000
20000
25000
30000
Iterations
Density of beta[2]
0e+00
-3.5e-06
-2.5e-06
-1.5e-06
-5.0e-07
N = 4000 Bandwidth = 9.861e-08
Trace of beta[3]
0.0004
15000
20000
25000
30000
Iterations
Density of beta[3]
0
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
N = 4000 Bandwidth = 2.528e-05
Trace of beta[4]
-1.0
15000
20000
25000
30000
Iterations
Density of beta[4]
0.0
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
N = 4000 Bandwidth = 0.03213
Trace of beta[5]
-1.0
15000
20000
25000
30000
Iterations
Density of beta[5]
0.0
-1.0
-0.5
0.0
0.5
N = 4000 Bandwidth = 0.0432
Trace of beta[6]
-0.05
15000
20000
25000
30000
Iterations
Density of beta[6]
0
-0.05
0.00
0.05
0.10
N = 4000 Bandwidth = 0.004852
Trace of beta[7]
-1.5
15000
20000
25000
30000
Iterations
Density of beta[7]
0.0
-1.5
-1.0
-0.5
0.0
0.5
N = 4000 Bandwidth = 0.05133
Trace of tausq.inv
0.5
15000
20000
25000
30000
Iterations
Density of tausq.inv
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
N = 4000 Bandwidth = 0.07228

```
tau_post = samples2[[1]][,8]
alpha_post = matrix(NA, nrow = s, ncol = length(tau_post))
for(i in 1 : s){
  alpha_post[i,] = rnorm(length(tau_post), 0, 1/sqrt(tau_post))
}
alpha_post_mean = rowMeans(alpha_post)

library(ggplot2)
library(viridis)
```

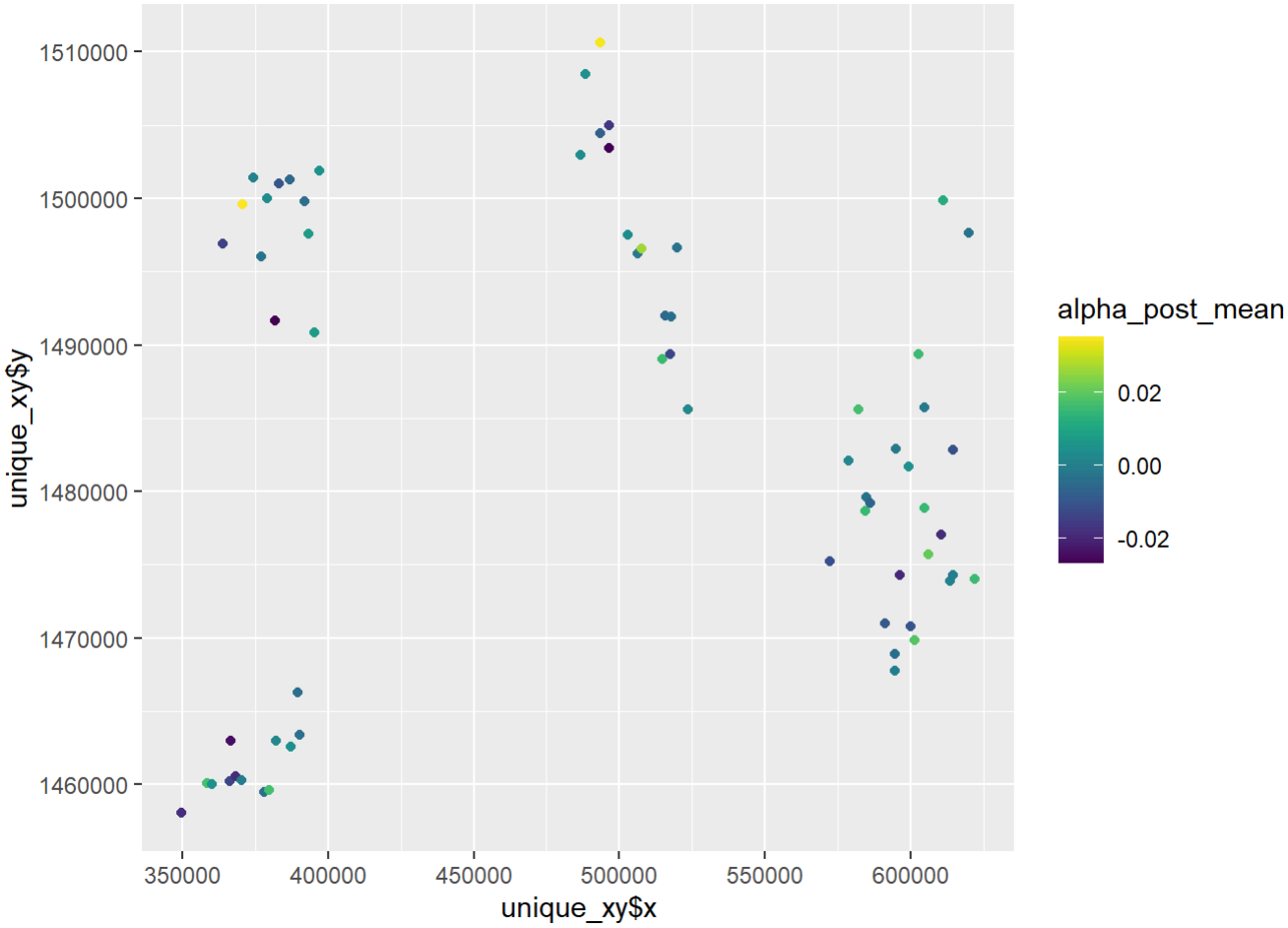
```
## Loading required package: viridisLite
```

```
ggplot()+
  geom_point(aes(x = unique_xy$x, y = unique_xy$y, col = alpha_post_mean))+
  scale_color_viridis()
```



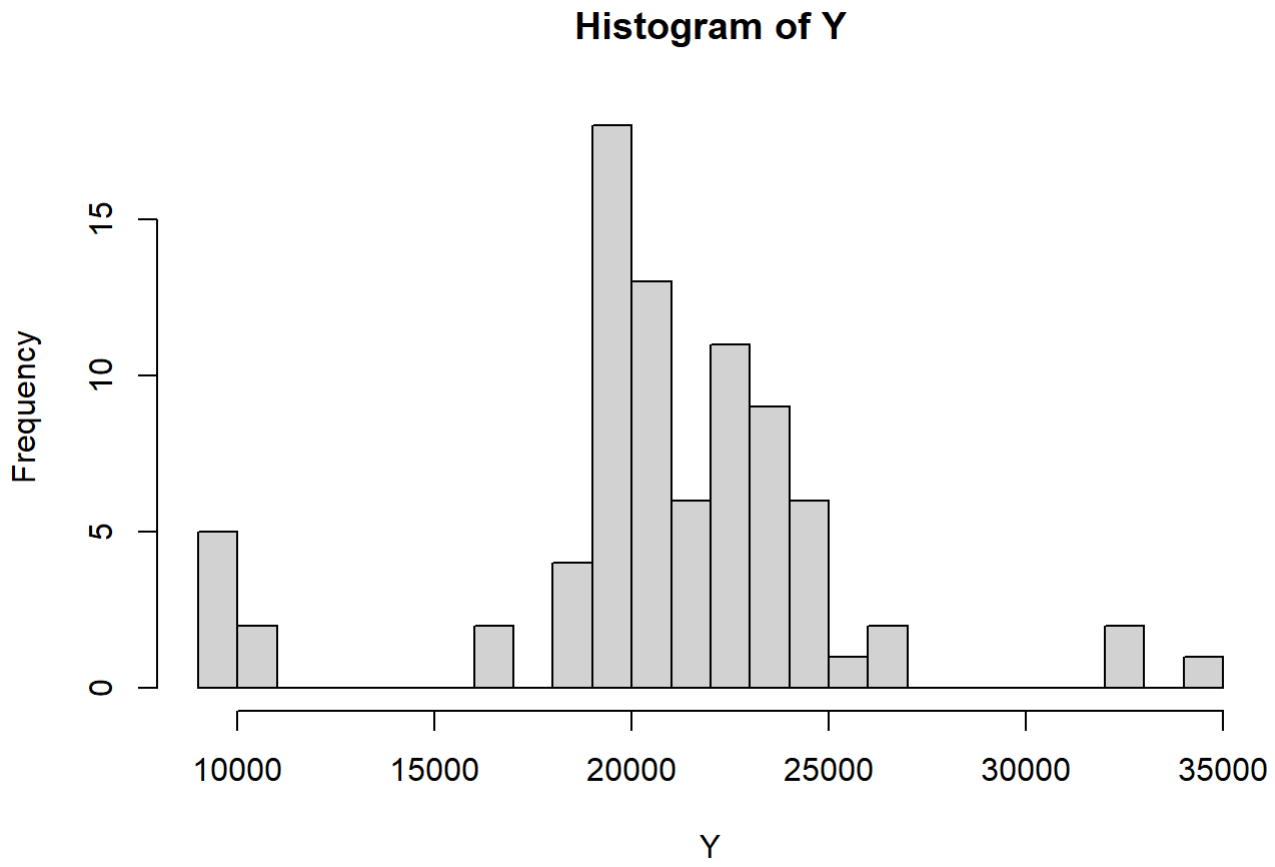
posterior means of the alphas by their spatial locations

```
tau_post = samples2[[1]][,8]
alpha_post = matrix(NA, nrow = s, ncol = length(tau_post))
for(i in 1 : s){
  alpha_post[i,] = rnorm(length(tau_post), 0, 1/sqrt(tau_post))
}
alpha_post_mean = rowMeans(alpha_post)

library(ggplot2)
library(viridis)
ggplot()+
  geom_point(aes(x = unique_xy$x, y = unique_xy$y, col = alpha_post_mean))+
  scale_color_viridis()
```

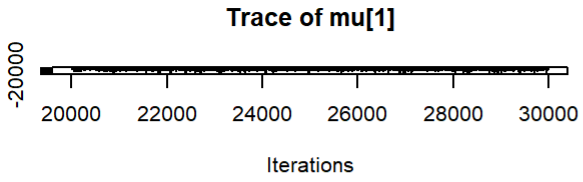


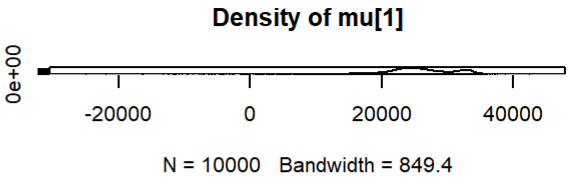
Question - 2

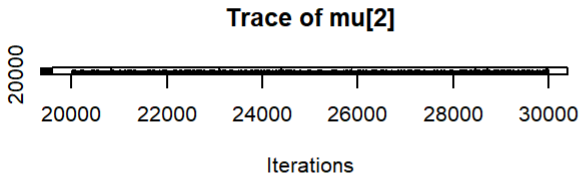
```
library(rjags)
library(ggplot2)
library(MASS)
data("galaxies")
Y = galaxies
hist(Y, breaks = 25)
```

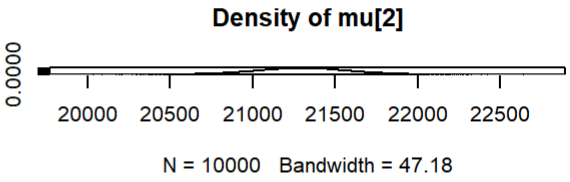


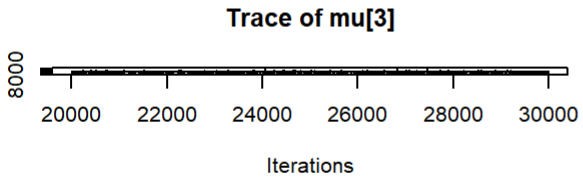

```
n = length(Y)

data = list(Y = Y, N = n, K = 3, alpha = rep(1, 3))

model_string = "model{

  # Likelihood
  for (i in 1:N) {
    Y[i] ~ dnorm(mu[Z[i]], tau[Z[i]])
    Z[i] ~ dcat(theta[])
  }
  for (j in 1:K) {
    mu[j] ~ dnorm(0, 1e-8)
    tau[j] ~ dgamma(0.01, 0.01)
  }

  theta[1:K] ~ ddirch(alpha[])

}"

params = c('mu', 'tau', 'theta')

model = jags.model(textConnection(model_string), data = data,
                   quiet = TRUE)
update(model, 2e4)
samples <- coda.samples(model, variable.names = params, n.iter = 1e4)
plot(samples)
```

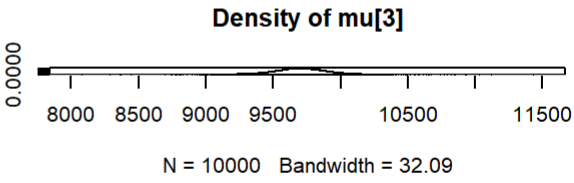

Trace of mu[1]
-20000
20000 22000 24000 26000 28000 30000
Iterations

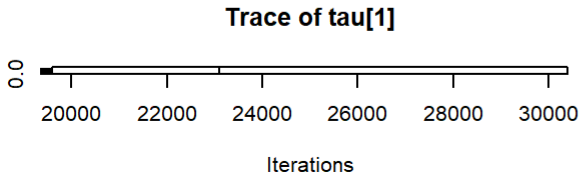

Density of mu[1]
0e+00
-20000 0 20000 40000
N = 10000 Bandwidth = 849.4

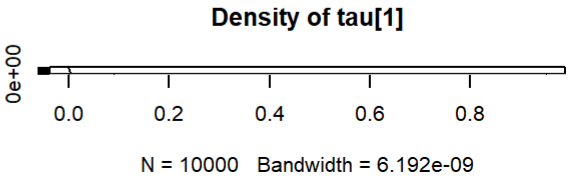

Trace of mu[2]
20000
20000 22000 24000 26000 28000 30000
Iterations

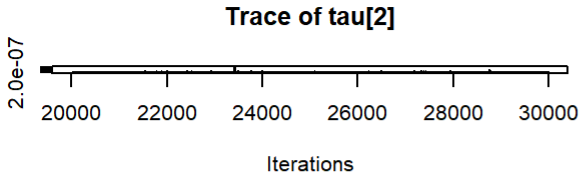

Density of mu[2]
0.0000
20000 20500 21000 21500 22000 22500
N = 10000 Bandwidth = 47.18

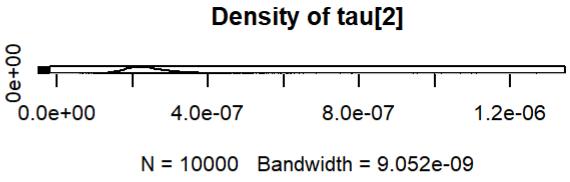

Trace of mu[3]
8000
20000 22000 24000 26000 28000 30000
Iterations

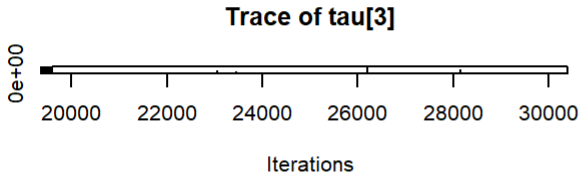

Density of mu[3]
0.0000
8000 8500 9000 9500 10500 11500
N = 10000 Bandwidth = 32.09

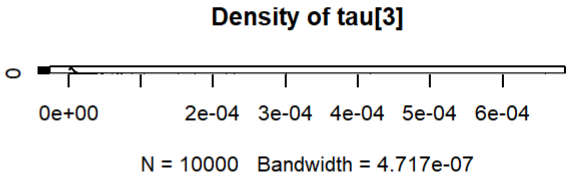

Trace of tau[1]
0.0
20000 22000 24000 26000 28000 30000
Iterations

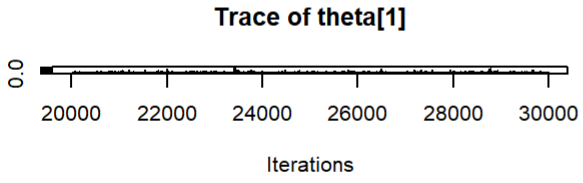

Density of tau[1]
0e+00
0.0 0.2 0.4 0.6 0.8
N = 10000 Bandwidth = 6.192e-09

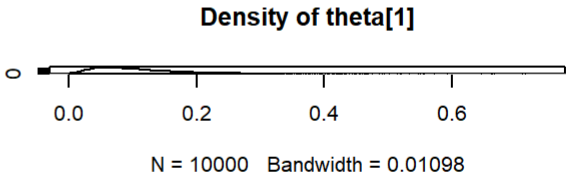

Trace of tau[2]
2.0e-07
20000 22000 24000 26000 28000 30000
Iterations

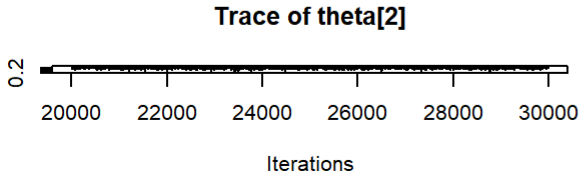

Density of tau[2]
0e+00
0.0e+00 4.0e-07 8.0e-07 1.2e-06
N = 10000 Bandwidth = 9.052e-09

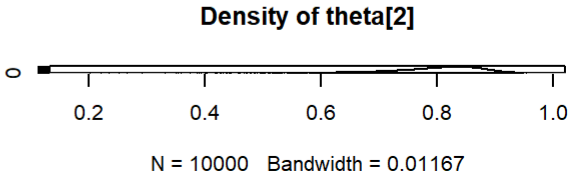

Trace of tau[3]
0e+00
20000 22000 24000 26000 28000 30000
Iterations

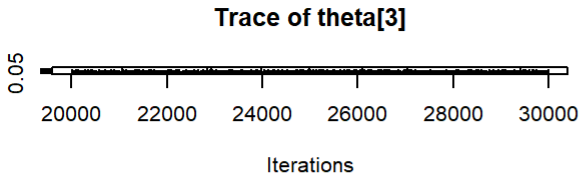

Density of tau[3]
0
0e+00 2e-04 3e-04 4e-04 5e-04 6e-04
N = 10000 Bandwidth = 4.717e-07

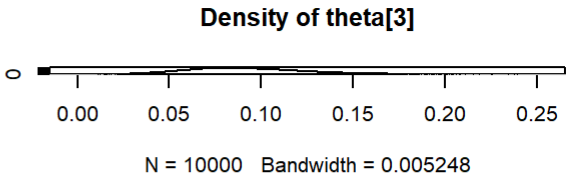

Trace of theta[1]
0.0
20000 22000 24000 26000 28000 30000
Iterations


Density of theta[1]
0
0.0 0.2 0.4 0.6
N = 10000 Bandwidth = 0.01098


Trace of theta[2]
0.2
20000 22000 24000 26000 28000 30000
Iterations


Density of theta[2]
0
0.2 0.4 0.6 0.8 1.0
N = 10000 Bandwidth = 0.01167

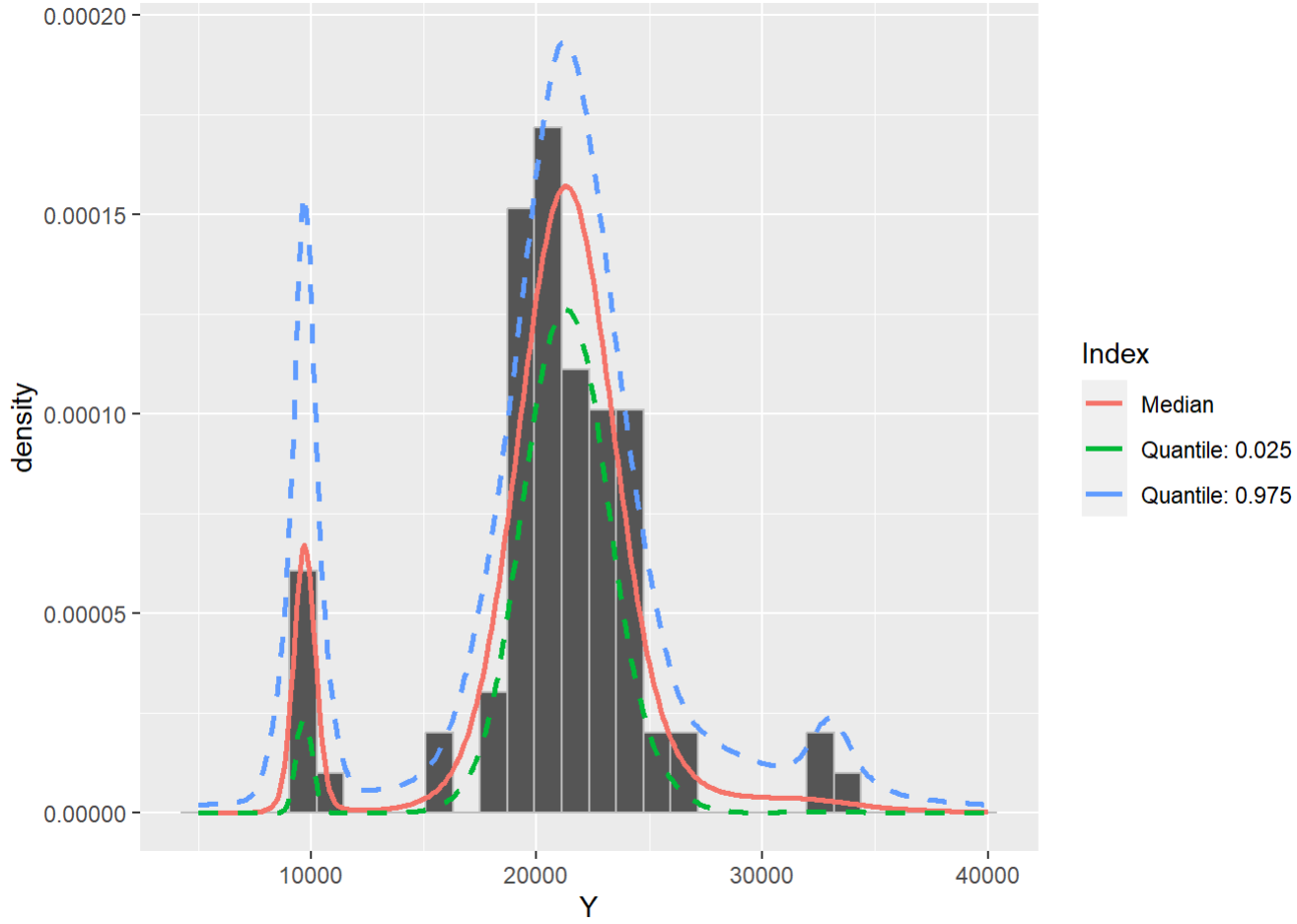

Trace of theta[3]
0.05
20000
22000
24000
26000
28000
30000
Iterations


Density of theta[3]
0
0.00
0.05
0.10
0.15
0.20
0.25
N = 10000 Bandwidth = 0.005248

```
y = seq(5e3, 4e4, 100)
S = 1e4
mu.post = samples[[1]][,1:3]
tau.post = samples[[1]][,4:6]
theta.post = samples[[1]][,7:9]


post_density = matrix(NA, nrow = S, ncol = 351)

for(i in 1:S){
  mu <- as.numeric(mu.post[i,])
  sigma <- as.numeric(1/sqrt(tau.post[i,]))
  theta <- as.numeric(theta.post[i, ])


  mix_gauss <- function(x) {
    theta[1] * dnorm(x, mean = mu[1], sd = sigma[1]) +
      theta[2] * dnorm(x, mean = mu[2], sd = sigma[2]) +
      theta[3] * dnorm(x, mean = mu[3], sd = sigma[3])
  }
  post_density[i, ] <- sapply(y, mix_gauss)
}

post_median <- apply(post_density, 2, median)
post_2.5.quantile <- apply(post_density, 2, quantile, probs = 0.025)
post_97.5.quantile <- apply(post_density, 2, quantile, probs = 0.975)


par(mfrow = c(1,1))
ggplot()+
  geom_histogram(aes(x = Y, y = after_stat(density)), col = 'grey')+
  geom_line(aes(y, post_median, col = 'Median'), size = 1)+
  geom_line(aes(y, post_2.5.quantile, col = 'Quantile: 0.025'), linetype = 'dashed', size =
1)+
  geom_line(aes(y, post_97.5.quantile, col = 'Quantile: 0.975'), linetype = 'dashed', size =
1)+
  labs(col = 'Index')
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



this mixture model does not fit the data well as the above density estimation does not visually match the density/histogram plot of Y as seen above

Question - 3

Calculating bayes factor

```
Y = c(563, 10)
N = c(2820, 27)

bf.c = function(c){
  p.y.m1 = pgamma(c, Y[1]+1, N[1], log.p = T) + pgamma(c, Y[2]+1, N[2], log.p = T) - log(c^2
* prod(N))
  p.y.m2  = lfactorial(sum(Y)) - sum(lfactorial(Y)) +
    sum(Y * log(N)) - (sum(Y) + 1) * log(sum(N)) +
    pgamma(c, sum(Y)+1, sum(N), log.p = T) - log(c)
  out = exp(p.y.m2 - p.y.m1)
  return(out)
}

bf.c(1)
```

```
## [1] 0.7155665
```

```
bf.c(10)
```

```
## [1] 7.154488
```

```
rm(list = ls())
library(rjags)
Y1 = 563
N1 = 2820
Y2 = 10
N2 = 27
```

```
## M1
model_string1 <- "model{
  lambda1 ~ dunif(0, 1)
  lambda2 ~ dunif(0, 1)
  Y1 ~ dpois(N1 * lambda1)
  Y2 ~ dpois(N2 * lambda2)
}"


# M2
model_string2 <- "
model {
  lmbdanot ~ dunif(0, 1)
  Y1 ~ dpois(N1 * lmbdanot)
  Y2 ~ dpois(N2 * lmbdanot)
}
"
data <- list(Y1 = Y1, N1 = N1, Y2 = Y2, N2 = N2)


model1 <- jags.model(textConnection(model_string1), data = data, n.chains = 1, quiet = TRUE)
update(model1, 10000, progress.bar = "none")
samps <- coda.samples(model1, variable.names = c("lambda1", "lambda2"),
                      n.iter = 20000, thin = 5, progress.bar = "none")
lambda1 <- samps[[1]][ , 1]
lambda2 <- samps[[1]][ , 2]


model2 <- jags.model(textConnection(model_string2), data = data, n.chains = 1, quiet = TRUE)
update(model2, 10000, progress.bar = "none")
samps <- coda.samples(model2, variable.names = "lmbdanot",
                      n.iter = 20000, thin = 5, progress.bar = "none")
lmbdanot <- samps[[1]][ , 1]


loglike.m1 <- sapply(1:4000, function(iter){
  dpois(Y1, N1*lambda1[iter] ,log = TRUE) + dpois(Y2, N2*lambda2[iter],log = TRUE)})

loglike.m2 <- sapply(1:4000, function(iter){
  dpois(Y1, N1*lmbdanot[iter] ,log = TRUE)+ dpois(Y2, N2*lmbdanot[iter],log = TRUE)})

deviance.m1 <- -2 * loglike.m1
deviance.m2 <- -2 * loglike.m2
```

calculating DIC for both the models

```
Dbar.m1 <- mean(deviance.m1)
Dbar.m2 <- mean(deviance.m2)

D.thetahat.m1 <- sum(dpois(Y1, N1*lambda1,log = TRUE ) + dpois(Y2, N2*lambda2,log = TRUE))
D.thetahat.m2 <- sum(dpois(Y1, N1*lmbdanot,log = TRUE )+ dpois(Y2, N2*lmbdanot ,log = TRUE))

pD.m1 <- Dbar.m1 - D.thetahat.m1
pD.m2 <- Dbar.m2 - D.thetahat.m2
DIC1.m1 <- pD.m1 + Dbar.m1
DIC1.m2 <- pD.m2 + Dbar.m2
DIC1.m1
```

```
## [1] 28754.36
```

calculating WAIC for both the models

```
posmeans.m1 <- mean(loglike.m1)
posmeans.m2 <- mean(loglike.m2)
posvars.m1 <- var(loglike.m1)
posvars.m2 <- var(loglike.m2)


WAIC1.m1 <- -2 * posmeans.m1 + 2 * posvars.m1
WAIC1.m2 <- -2 * posmeans.m2 + 2 *posvars.m2
WAIC1.m1
```

```
## [1] 16.34523
```

For c = 10

```
## M1
model_string1 <- "model{
  lambda1 ~ dunif(0, 10)
  lambda2 ~ dunif(0, 10)
  Y1 ~ dpois(N1 * lambda1)
  Y2 ~ dpois(N2 * lambda2)
}"


# M2
model_string2 <- "
model {
  lmbdanot ~ dunif(0, 10)
  Y1 ~ dpois(N1 * lmbdanot)
  Y2 ~ dpois(N2 * lmbdanot)
}
"
data <- list(Y1 = Y1, N1 = N1, Y2 = Y2, N2 = N2)

model1 <- jags.model(textConnection(model_string1), data = data, n.chains = 1, quiet = TRUE)
update(model1, 10000, progress.bar = "none")
samps <- coda.samples(model1, variable.names = c("lambda1", "lambda2"),
                      n.iter = 20000, thin = 5, progress.bar = "none")
lambda1 <- samps[[1]][ , 1]
lambda2 <- samps[[1]][ , 2]

model2 <- jags.model(textConnection(model_string2), data = data, n.chains = 1, quiet = TRUE)
update(model2, 10000, progress.bar = "none")
samps <- coda.samples(model2, variable.names = "lmbdanot",
                      n.iter = 20000, thin = 5, progress.bar = "none")
lmbdanot <- samps[[1]][ , 1]

loglike.m1 <- sapply(1:4000, function(iter){
  dpois(Y1, N1*lambda1[iter] ,log = TRUE) + dpois(Y2, N2*lambda2[iter],log = TRUE)})

loglike.m2 <- sapply(1:4000, function(iter){
  dpois(Y1, N1*lmbdanot[iter] ,log = TRUE)+ dpois(Y2, N2*lmbdanot[iter],log = TRUE)})

deviance.m1 <- -2 * loglike.m1
deviance.m2 <- -2 * loglike.m2

# DIC
Dbar.m1 <- mean(deviance.m1)
Dbar.m2 <- mean(deviance.m2)

D.thetahat.m1 <- sum(dpois(Y1, N1*lambda1,log = TRUE ) + dpois(Y2, N2*lambda2,log = TRUE))
D.thetahat.m2 <- sum(dpois(Y1, N1*lmbdanot,log = TRUE )+ dpois(Y2, N2*lmbdanot ,log = TRUE))

pD.m1 <- Dbar.m1 - D.thetahat.m1
pD.m2 <- Dbar.m2 - D.thetahat.m2
DIC10.m1 <- pD.m1 + Dbar.m1
DIC10.m2 <- pD.m2 + Dbar.m2
DIC10.m1
```

```
## [1] 28677.76
```

```
DIC10.m2
```

```
## [1] 32833.03
```

```
# WAIC

posmeans.m1 <- mean(loglike.m1)
posmeans.m2 <- mean(loglike.m2)
posvars.m1 <- var(loglike.m1)
posvars.m2 <- var(loglike.m2)


WAIC10.m1 <- -2 * posmeans.m1 + 2 * posvars.m1
WAIC10.m2 <- -2 * posmeans.m2 + 2 *posvars.m2
WAIC10.m1
```

```
## [1] 16.33964
```

Comparing DIC and WAIC values for c=1 and c= 10

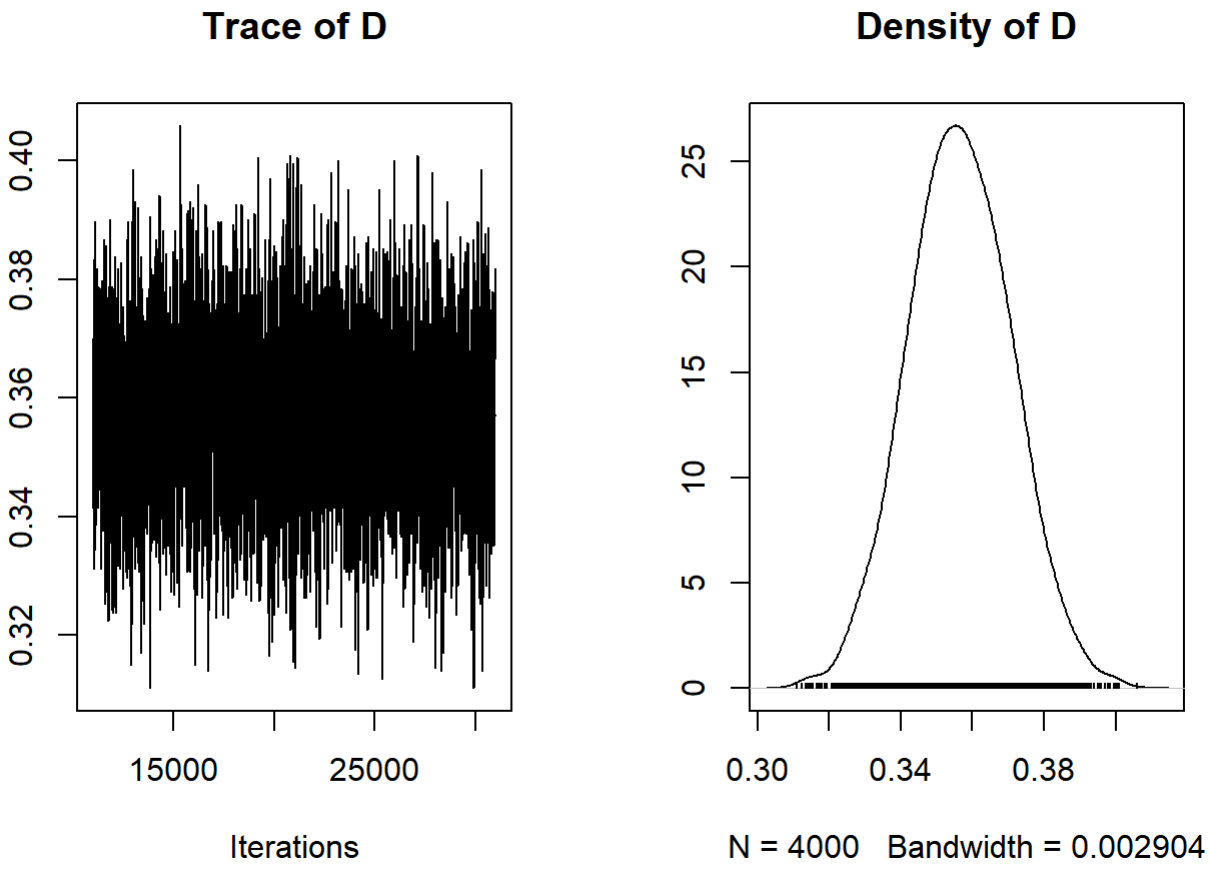
```
# Create a data frame
comparison_table <- data.frame(
  c = c(1, 10),
  DIC = c(DIC1.m1, DIC10.m1),
  WAIC = c(WAIC1.m1, WAIC10.m1)
)

knitr::kable(comparison_table, caption = "Comparison of DIC and WAIC for different values of
c")
```

Comparison of DIC and WAIC for different values of c

c	DIC	WAIC
1	28754.36	16.34523
10	28677.76	16.33964

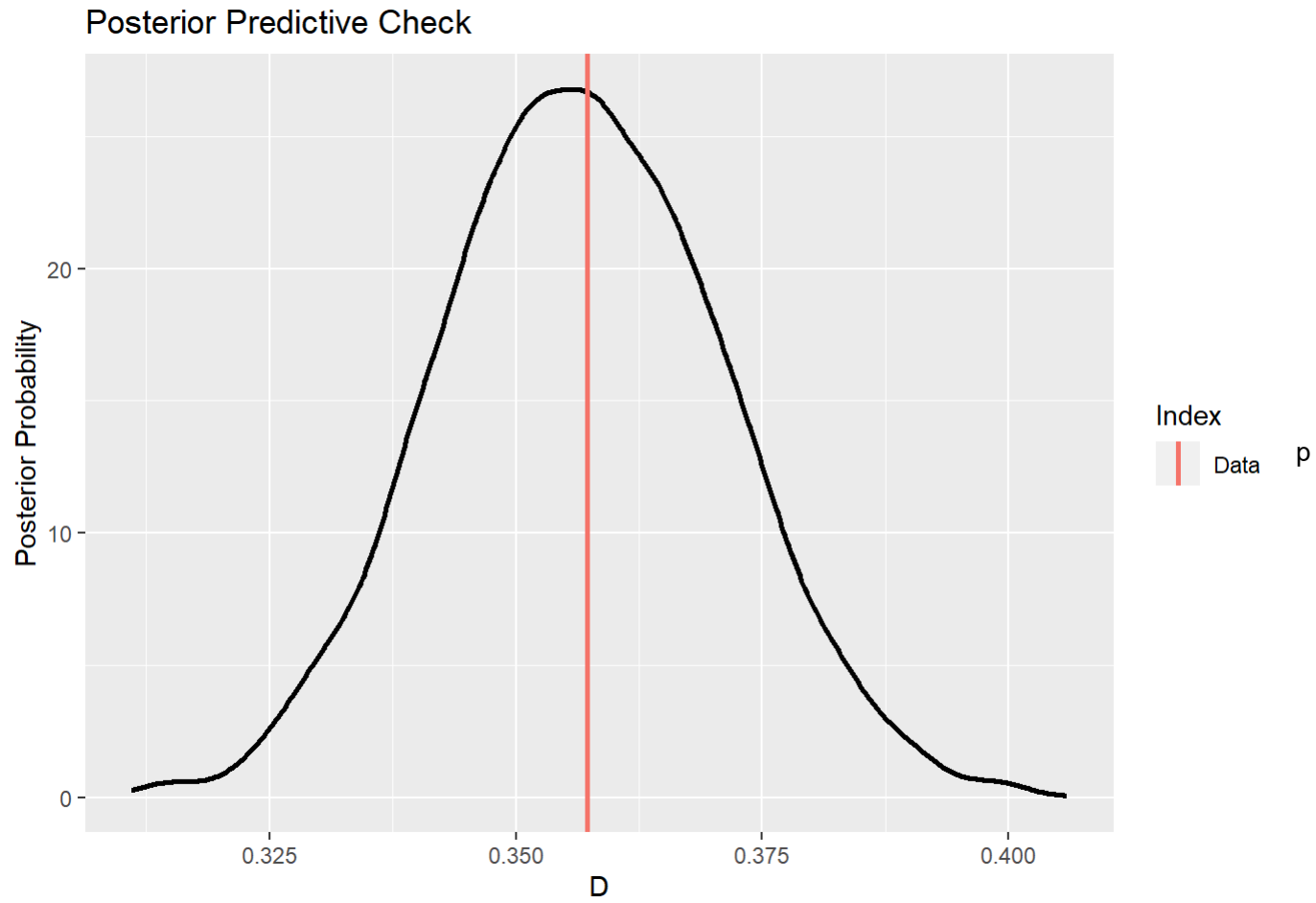
Question - 4

```
library(geoR)
data("gambia")

Y = gambia$pos
X = gambia[,-3]

n = length(Y)
p = ncol(X)

library(rjags)

data = list(Y = Y, n = n, p = p, X = X)

params = c('D')

model_string_1 = textConnection("model{
      #Likelihood
      for(i in 1:n){
      Y[i] ~ dbinom(probs[i], 1)
      logit(probs[i]) = inprod(X[i,], beta)
      }
      #Priors
      for(j in 1:p){
      beta[j] ~ dnorm(0, 1e-3)
      }
      # Posterior preditive checks
      for(i in 1:n){
      Y1[i] ~ dbinom(probs[i], 1)
      }
      D = mean(Y1[])
}")



model = jags.model(model_string_1, data = data,
                     quiet = TRUE)
update(model, 1e4)

samples = coda.samples(model, variable.names = params,
                         thin = 5, n.iter = 2e4)
plot(samples)
```




```
D0 = mean(Y)
D = samples[[1]]
pval = mean(D > D0)
library(ggplot2)

ggplot()+
  geom_density(aes(x = D, y = after_stat(density)), size = 1)+
  geom_vline(aes(xintercept = D0, col = 'Data'), size = 1)+
  labs(col = 'Index',
       title = 'Posterior Predictive Check',
       y = 'Posterior Probability')
```

```
## Don't know how to automatically pick scale for object of type <mcmc>.
## Defaulting to continuous.
```



value near 0 or 1 indicates bad fit

Question - 5

```
library(datasets)
data("WWWusage")

Y = WWWusage
n = length(WWWusage)
data = list(Y=Y, n=n)
model_string.1 = textConnection("model{

  # L = 1

  for(t in 5:n){
    mu[t] = beta[1] + beta[2] * Y[t-1]
    Y[t] ~ dnorm(mu[t], tau)
  }

  beta[1] ~ dnorm(0, 1e-4)
  beta[2] ~ dnorm(0, 1e-4)
  tau ~ dgamma(0.1, 0.1)
  sigma = sqrt(1/tau)
}")

m1 = jags.model(model_string.1, data=data, n.chains=1, quiet=T)
update(m1, 1e4, progress.bar="none")
samp1 = coda.samples(m1, variable.names=c("beta", "sigma"), n.iter=2e4, thin=5)

model_string.2 = textConnection("model{

  # L = 2

  for(t in 5:n){
    mu[t] = beta[1] + beta[2] * Y[t-1] + beta[3] * Y[t-2]
    Y[t] ~ dnorm(mu[t], tau)
  }

  beta[1] ~ dnorm(0, 1e-4)
  beta[2] ~ dnorm(0, 1e-4)
  beta[3] ~ dnorm(0, 1e-4)
  tau ~ dgamma(0.1, 0.1)
  sigma = sqrt(1/tau)
}")

m2 = jags.model(model_string.2, data=data, n.chains=1, quiet=T)
update(m2, 1e4, progress.bar="none")
samp2 = coda.samples(m2, variable.names=c("beta", "sigma"), n.iter=2e4, thin=5)

model_string.3 = textConnection("model{
  # L = 3

  for(t in 5:n){
    mu[t] = beta[1] + beta[2] * Y[t-1] + beta[3] * Y[t-2] + beta[4] * Y[t-3]
    Y[t] ~ dnorm(mu[t], tau)
  }
```

```
  beta[1] ~ dnorm(0, 1e-4)
  beta[2] ~ dnorm(0, 1e-4)
  beta[3] ~ dnorm(0, 1e-4)
  beta[4] ~ dnorm(0, 1e-4)
  tau ~ dgamma(0.1, 0.1)
  sigma = sqrt(1/tau)
}")

m3 = jags.model(model_string.3, data=data, n.chains=1, quiet=T)
update(m3, 1e4, progress.bar="none")
samp3 = coda.samples(m3, variable.names=c("beta", "sigma"), n.iter=2e4, thin=5)

model_string.4 = textConnection("model{

  # L = 4

  for(t in 5:n){
    mu[t] = beta[1] + beta[2] * Y[t-1] + beta[3] * Y[t-2] + beta[4] * Y[t-3] + beta[5] * Y[t-
4]
    Y[t] ~ dnorm(mu[t], tau)
  }

  beta[1] ~ dnorm(0, 1e-4)
  beta[2] ~ dnorm(0, 1e-4)
  beta[3] ~ dnorm(0, 1e-4)
  beta[4] ~ dnorm(0, 1e-4)
  beta[5] ~ dnorm(0, 1e-4)
  tau ~ dgamma(0.1, 0.1)
  sigma = sqrt(1/tau)
}")

m4 = jags.model(model_string.4, data=data, n.chains=1, quiet=T)
update(m4, 1e4, progress.bar="none")
samp4 = coda.samples(m4, variable.names=c("beta", "sigma"), n.iter=2e4, thin=5)
```

Comparing model using WAIC

```
post_samples.num = nrow(samp1[[1]])
log_likelihood_vector = function(beta, sigma){
  l = length(beta)
  ans = sapply(5:n, function(i){
    mu = sum(beta[2:l] * Y[(i-1):(i-l+1)])
    t = dnorm(Y[i], mu, sigma, log=T)
    return(t)
  })
  return(ans)
}

log_lik.m1 = sapply(1:post_samples.num, function(r) log_likelihood_vector(samp1[[1]][r, 1:2],
samp1[[1]][r, 3]))
log_lik.m2 = sapply(1:post_samples.num, function(r) log_likelihood_vector(samp2[[1]][r, 1:3],
samp2[[1]][r, 4]))
log_lik.m3 = sapply(1:post_samples.num, function(r) log_likelihood_vector(samp3[[1]][r, 1:4],
samp3[[1]][r, 5]))
log_lik.m4 = sapply(1:post_samples.num, function(r) log_likelihood_vector(samp4[[1]][r, 1:5],
samp4[[1]][r, 6]))

pos_means.m1 = apply(log_lik.m1, 1, mean)
pos_means.m2 = apply(log_lik.m2, 1, mean)
pos_means.m3 = apply(log_lik.m3, 1, mean)
pos_means.m4 = apply(log_lik.m4, 1, mean)

pos_var.m1 = apply(log_lik.m1, 1, var)
pos_var.m2 = apply(log_lik.m2, 1, var)
pos_var.m3 = apply(log_lik.m3, 1, var)
pos_var.m4 = apply(log_lik.m4, 1, var)

WAIC.m1 = -2*sum(pos_means.m1) + 2*sum(pos_var.m1)
WAIC.m2 = -2*sum(pos_means.m2) + 2*sum(pos_var.m2)
WAIC.m3 = -2*sum(pos_means.m3) + 2*sum(pos_var.m3)
WAIC.m4 = -2*sum(pos_means.m4) + 2*sum(pos_var.m4)
```

we get lowest WAIC for m3: optimal L = 3