Assignment 5

Madhur Bansal (210572)

2024-04-15

Contents

Q1 (Ch: 4-8) **2**

a) Logistic Regression without Random Effects . . . 2

JAGS code . . . 3

Conclusion . . . 5

b) Logistic Regression with Random Effects . . . 6

Assigning Groups . . . 6

JAGS code . . . 6

Why Random Effects? . . . 9

Spatial Plot . . . 10

Q2 (Ch: 4-10) **11**

JAGS code . . . 11

Posterior density plot . . . 16

Conclusion . . . 16

Q3 (Ch: 5-4) **17**

Bayes Factor . . . 17

DIC and WAIC (c = 1) . . . 17

JAGS code . . . 17

DIC . . . 19

WAIC . . . 20

DIC and WAIC (c = 10) . . . 22

Conclusion . . . 23

Q4 (Ch: 5-6) **23**

JAGS code . . . 24

PPD checks . . . 25

Conclusion . . . 26

Q5 (Ch: 5-10) **27**

JAGS code 27

Different values of L 28

$L = 1$ 28

$L = 2$ 30

$L = 3$ 31

$L = 4$ 32

Conclusion 33

Q1 (Ch: 4-8)

We have to fit a logistic regression model using Y_i as the response variable and the other seven variables as covariates.

```
library(geoR)
library(rjags)
library(ggplot2)

### Preparing the data
data("gambia")
Y <- gambia$pos

# Added column of 1's for beta_0
X <- as.matrix(subset(x = gambia, select = -c(pos)))
X <- scale(X)

print(head(gambia))
```

```
##             x       y pos  age netuse treated green phc
## 1850 349631.3 1458055   1 1783      0       0 40.85   1
## 1851 349631.3 1458055   0  404      1       0 40.85   1
## 1852 349631.3 1458055   0  452      1       0 40.85   1
## 1853 349631.3 1458055   1  566      1       0 40.85   1
## 1854 349631.3 1458055   0  598      1       0 40.85   1
## 1855 349631.3 1458055   1  590      1       0 40.85   1
```

a) Logistic Regression without Random Effects

In this part, we fit the following model:

$$logit[Prob(Y_i = 1)] = \Sigma_{j=1}^{p} X_{ij}\beta_j$$

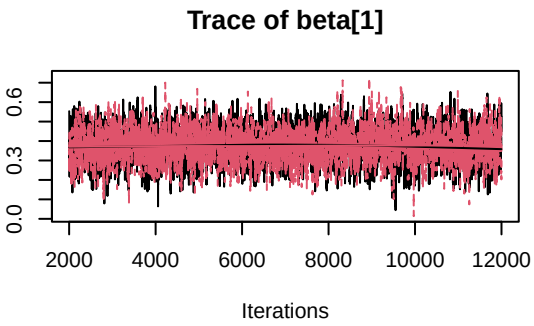
where Y_i is the response variable, X_i's are the covariates, and β_j are the regression coeff.

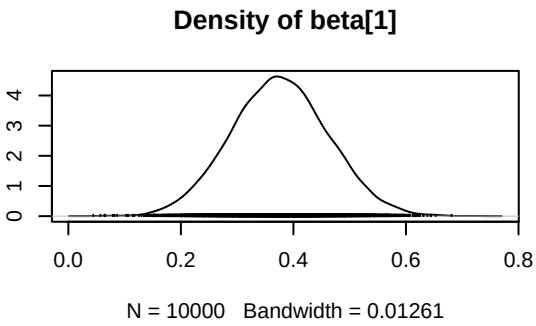
Note: Since it was taking a long time to run the MCMC chain each time, I have saved the samples from earlier run in an .Rdata file and I am using them here.

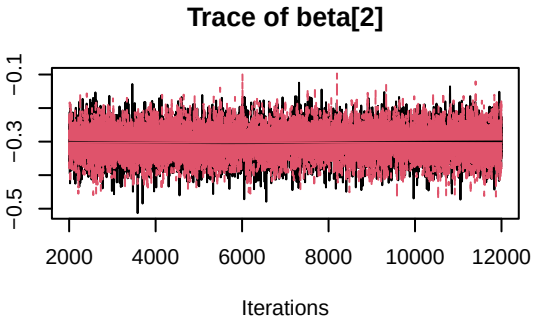
JAGS code

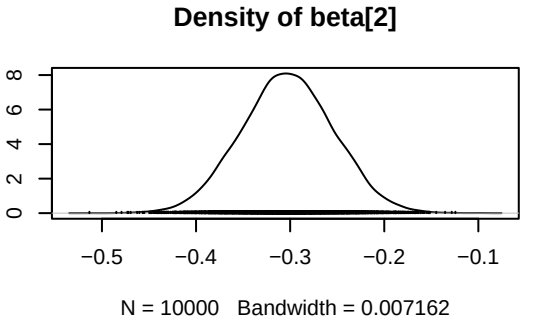
```
### JAGS model
modelString <- textConnection("model{

  # Likelihood
  for(i in 1:n)
  {
    Y[i] ~ dbern(q[i])
    logit(q[i]) = inprod(X[i, ], beta[])
  }

  # Priors
  for(j in 1:p)
  {
    beta[j] ~ dnorm(0, 100^(-2))
  }

}")

data.list <- list(Y = Y, X = X,
                  n = length(Y),
                  p = length(X[1, ]))
model <- jags.model(file = modelString,
                    data = data.list,
                    n.chains = 2)

# Burn-in
update(model, 1e3)

# Samples

n.samples <- 1e4
params <- c("beta")

samples <- coda.samples(model = model,
                        variable.names = params,
                        n.iter = n.samples)

# outA <- list(model = model,
#              samples = samples)
# save(outA, file = c('out-4A.Rdata'))
```

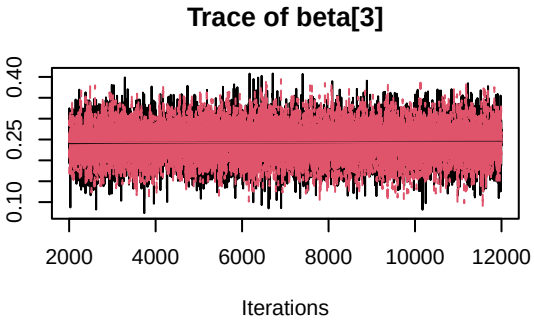
```
load('./out-4A.Rdata')
model <- outA$model
samples <- outA$samples

plot(samples)
```

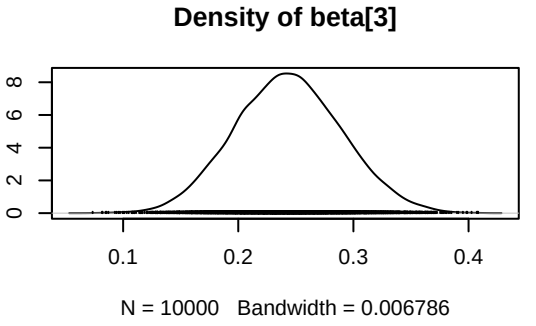

Trace of beta[1]
0.0
0.3
0.6
2000
4000
6000
8000
10000
12000
Iterations

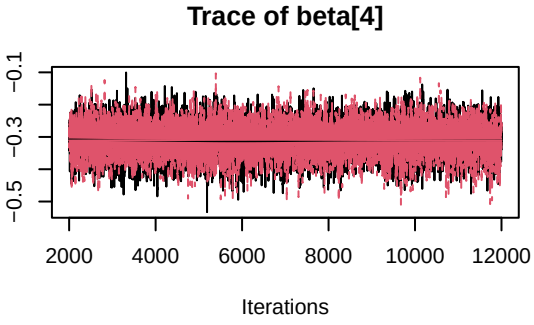

Density of beta[1]
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
N = 10000 Bandwidth = 0.01261

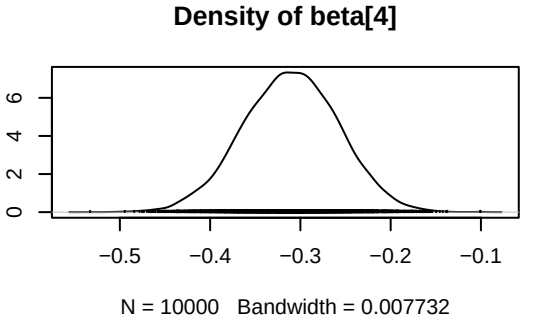

Trace of beta[2]
−0.5
−0.3
−0.1
2000
4000
6000
8000
10000
12000
Iterations

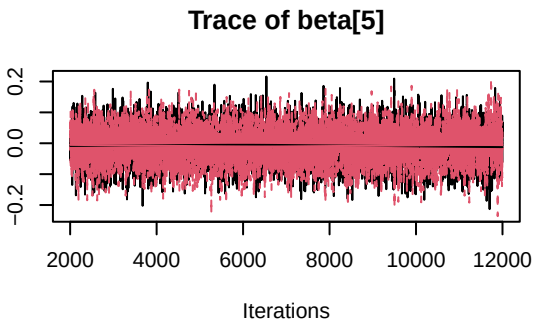

Density of beta[2]
0
2
4
6
8
−0.5
−0.4
−0.3
−0.2
−0.1
N = 10000 Bandwidth = 0.007162

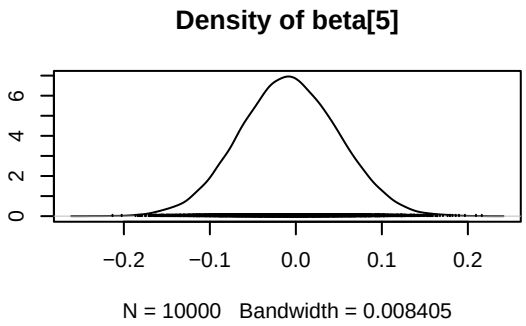

Trace of beta[3]
0.10
0.25
0.40
2000
4000
6000
8000
10000
12000
Iterations

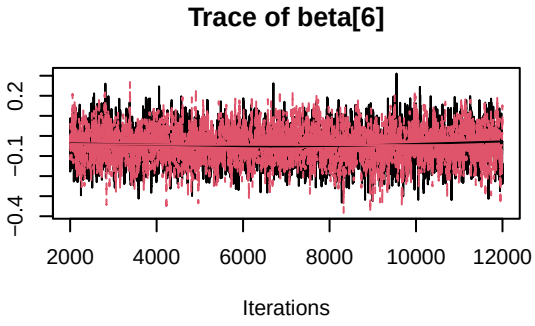

Density of beta[3]
0
2
4
6
8
0.1
0.2
0.3
0.4
N = 10000 Bandwidth = 0.006786

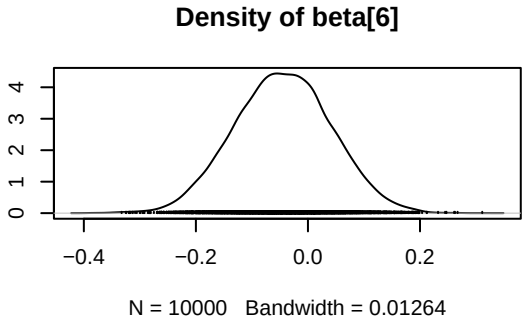

Trace of beta[4]
−0.5
−0.3
−0.1
2000
4000
6000
8000
10000
12000
Iterations

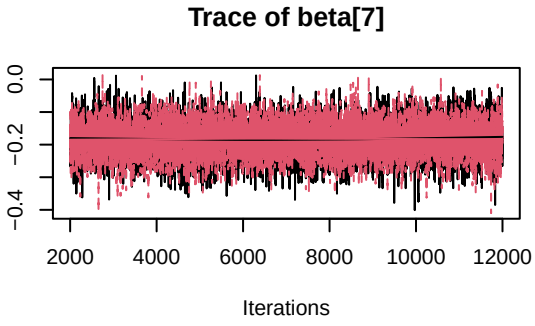

Density of beta[4]
0
2
4
6
−0.5
−0.4
−0.3
−0.2
−0.1
N = 10000 Bandwidth = 0.007732

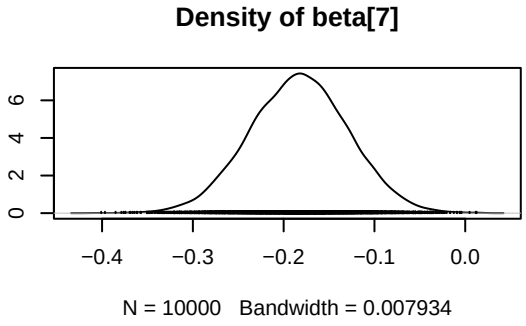












Conclusion

From the plots we can conclude that the MCMC chains have converged.

From the posterior density plots of beta, we can make the following conclusions:

- beta[5] and beta[6] are centered at zero, indicating **treated** and **green** are not significant.
- beta[3] > 0, indicates that the an **older** child is **more likely** to get infected.
- beta[4] < 0, indicates that **use** of **bed-net reduces** the chance of getting infected.
- beta[7] < 0, indicates that **presence of health-center** in village helps in **reducing** the active malaria cases.

b) Logistic Regression with Random Effects

In this part, we fit the following model:

$$logit[Prob(Y_i = 1)] = \Sigma_{j=1}^{p} X_{ij}\beta_j + \alpha_{s_i}$$

where α_l is the random effect based on location and $\alpha_l \sim N(0, \tau^2)$ (iid)

Assigning Groups

To assign groups (1, 2, ..., L) to the observations, I have created a vector **s**, that stores the group to which each observation belong.

```
unique.coords <- unique(x = cbind(gambia$x, gambia$y))
L <- length(unique.coords[,1])
s <- numeric(length = length(Y))

for (i in 1:L)
{
  coords <- unique.coords[i, ]
  index <- (gambia$x == coords[1] & gambia$y == coords[2])
  s[index] <- i
}

print(head(cbind(s, gambia)))
```

```
##      s        x       y pos  age netuse treated green phc
## 1850 1 349631.3 1458055   1 1783      0       0 40.85   1
## 1851 1 349631.3 1458055   0  404      1       0 40.85   1
## 1852 1 349631.3 1458055   0  452      1       0 40.85   1
## 1853 1 349631.3 1458055   1  566      1       0 40.85   1
## 1854 1 349631.3 1458055   0  598      1       0 40.85   1
## 1855 1 349631.3 1458055   1  590      1       0 40.85   1
```

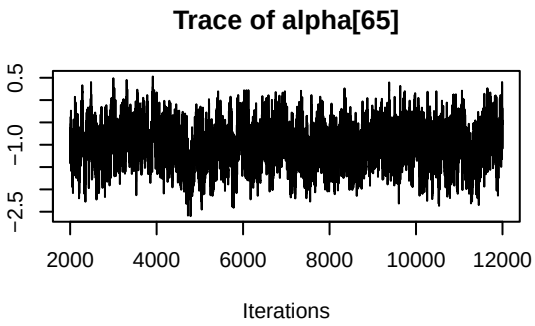
JAGS code

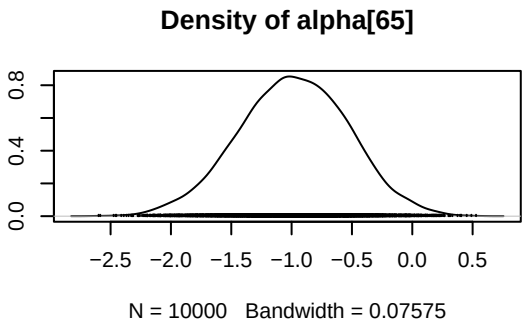
Note: Since it was taking a long time to run the MCMC chain each time, I have saved the samples from earlier run in an .Rdata file and I am using them here.

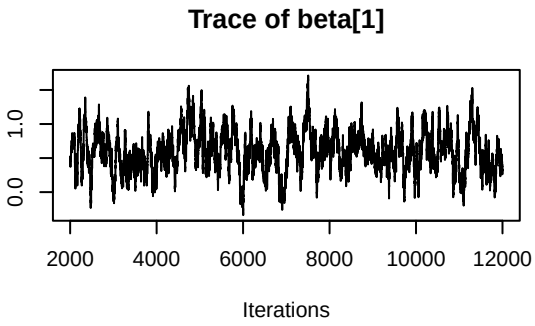
```
### JAGS model
modelString <- textConnection("model{

  # Likelihood
```

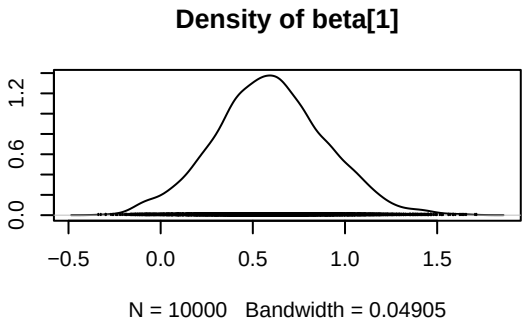
```
  for(i in 1:n)
  {
    Y[i] ~ dbern(q[i])
    logit(q[i]) = inprod(X[i, ], beta[]) + alpha[s[i]]
  }

  # Priors
  for(j in 1:L)
  {
    alpha[j] ~ dnorm(0, alpha.precision)
  }
  alpha.precision ~ dgamma(0.01, 0.01)
  tau.sq = 1/alpha.precision

  for(j in 1:p)
  {
    beta[j] ~ dnorm(0, 100^(-2))
  }

}")

data.list <- list(Y = Y,
                  X = X,
                  s = s,
                  n = length(Y),
                  p = length(X[1,]),
                  L = L)
model <- jags.model(file = modelString,
                    data = data.list,
                    n.chains = 2)

# Burn-in
update(model, 1e3)

# Samples
n.samples <- 1e4
params <- c("beta", "alpha")

samples <- coda.samples(model = model,
                        variable.names = params,
                        n.iter = n.samples)
```

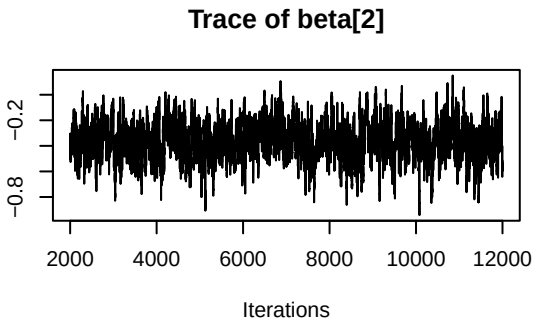
```
load('./out-4B.Rdata')
model <- outB$model
samples <- outB$samples

plot(samples[[1]][,L:(L+3)])
```

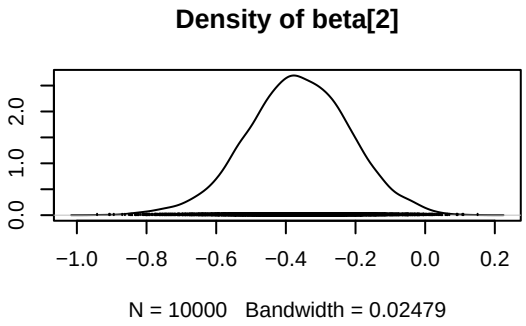


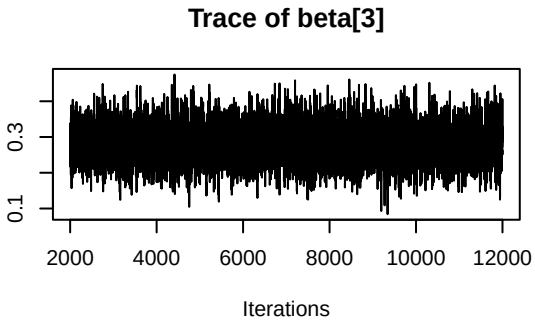


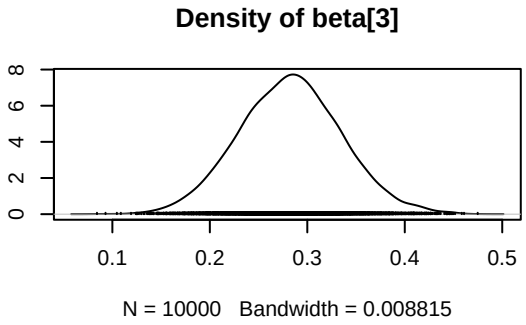




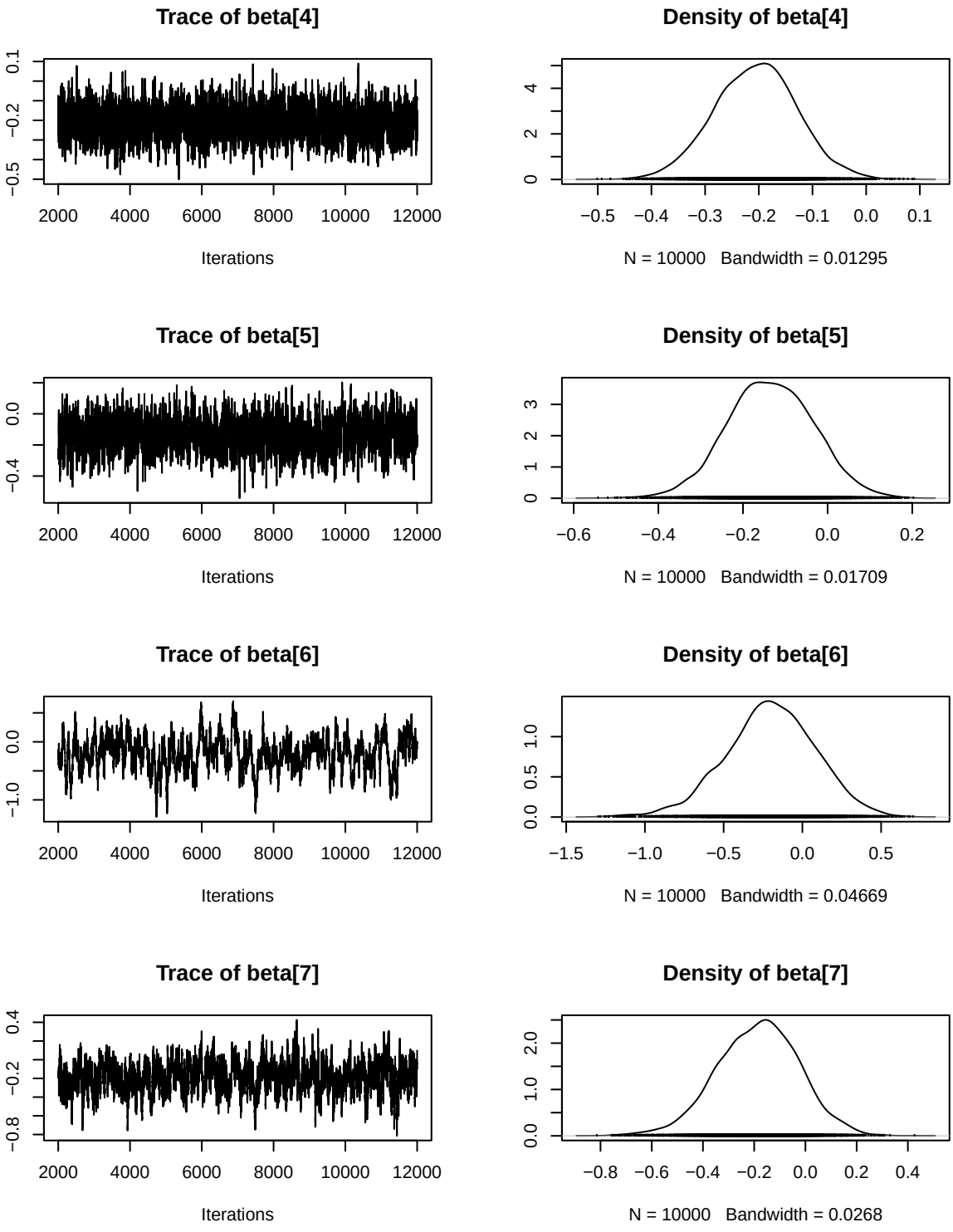









```
plot(samples[[1]][,(L+4):(L+7)])
```



From the plots, we can conclude that the MCMC chains have converged.

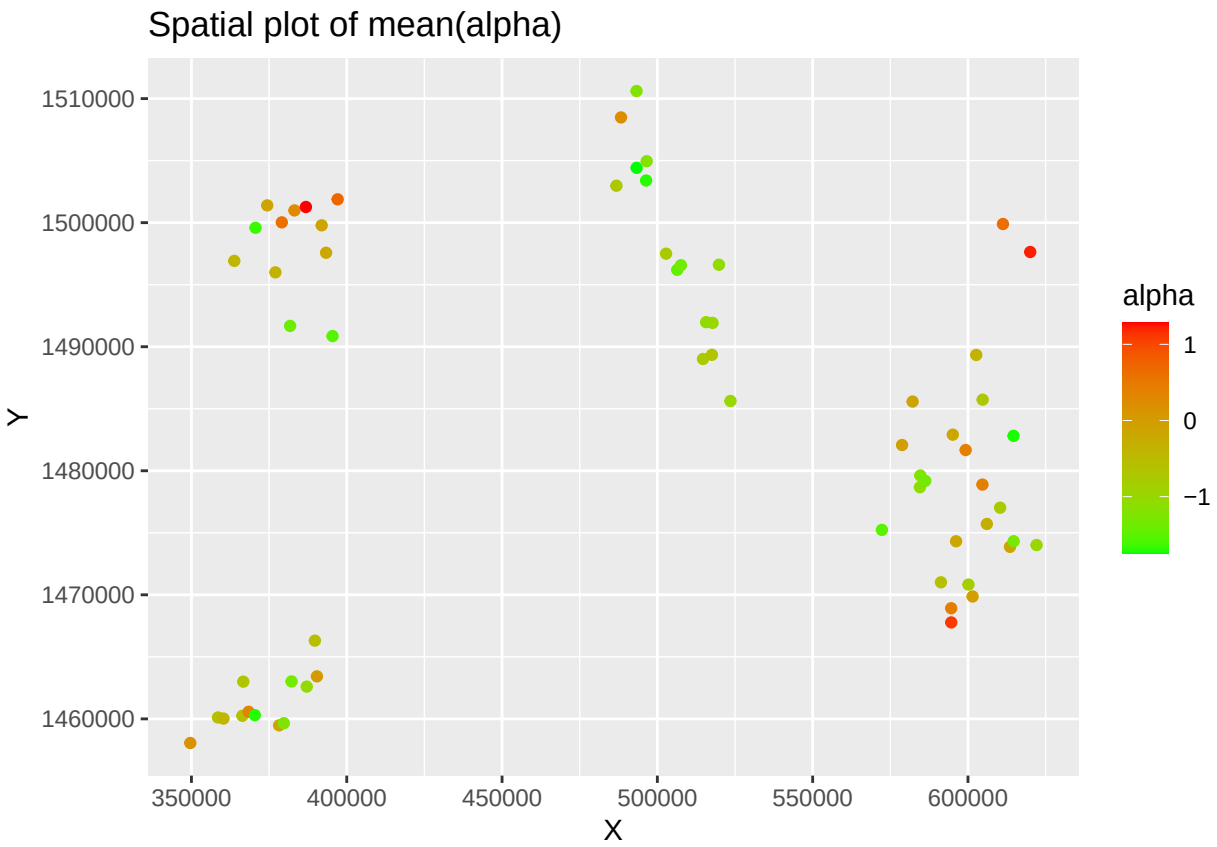
Why Random Effects?

Whether a child will test positive for malaria or not will depend on his/her surroundings:

- If there are already a lot of cases in an area, the chance of someone getting infected also increases.
- There may be more musquitos in a particular area, which will also increase the infection.

Spatial Plot

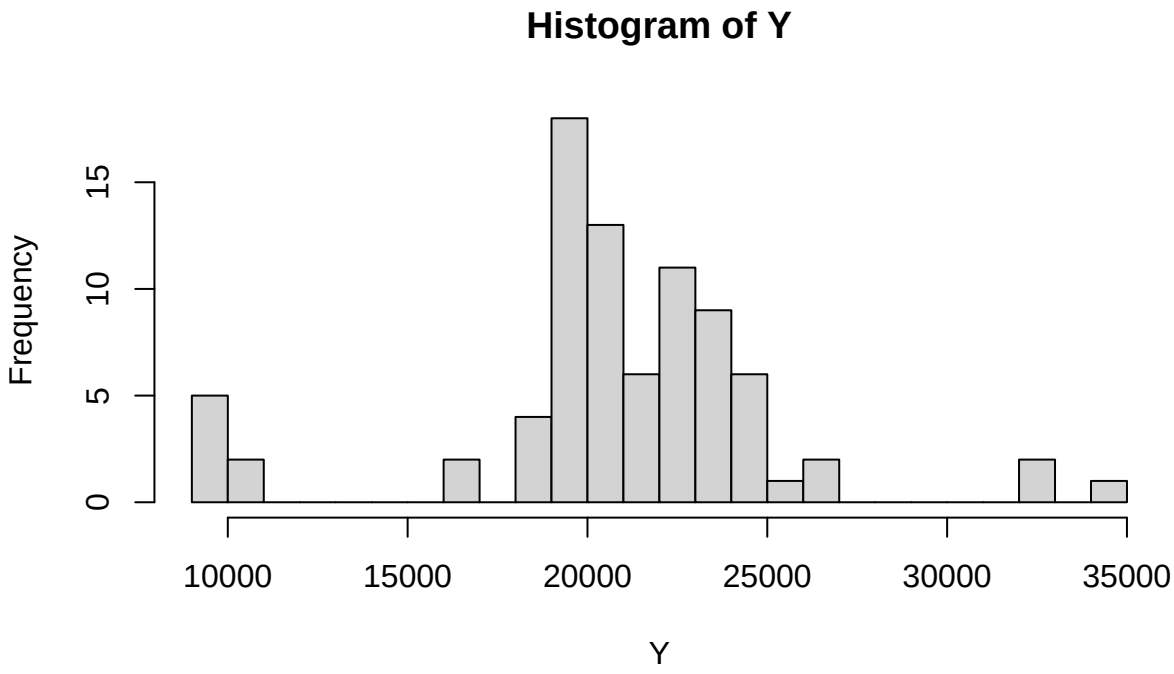
```
samples.mat <- as.matrix(samples[[1]][,1:L])
samples.mean <- apply(samples.mat, 2, mean)

par(mfrow = c(1,1))

plot.data <- data.frame(X = unique.coords[ ,1],
                        Y = unique.coords[ ,2],
                        alpha = samples.mean)
plt <- ggplot(plot.data, aes(X, Y))+
  geom_point(aes(color = alpha)) +
  scale_color_gradient(low = "green", high = "red") +
  labs(title = "Spatial plot of mean(alpha)")
plt
```



From the plot we can locate the areas where more Malaria cases are present. The **more** the value of **alpha**, the higher are the chances of finding an infected child.

Q2 (Ch: 4-10)

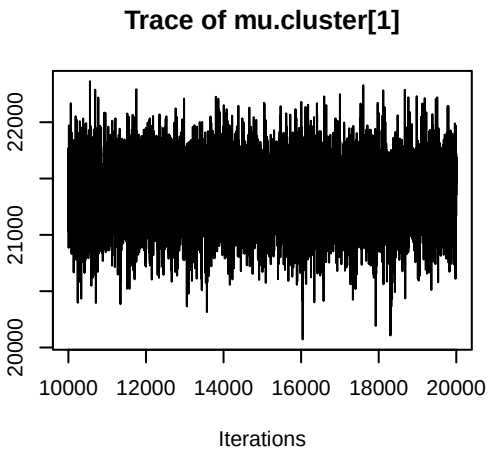
The objective of the question is to fit a mixture of K (= 3) normal distribution to the given observations $Y_1, Y_2, ..., Y_{82}$.

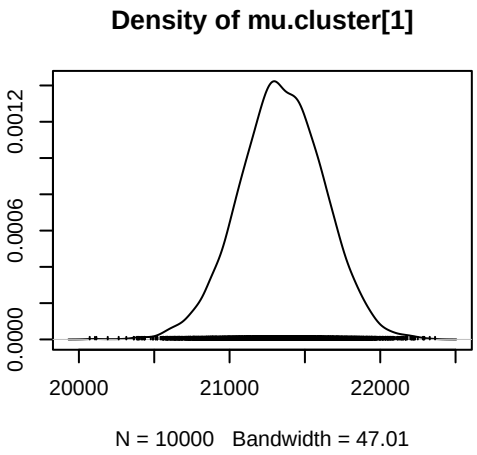


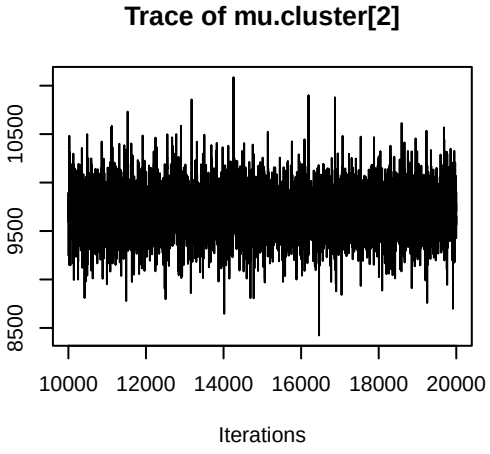
JAGS code

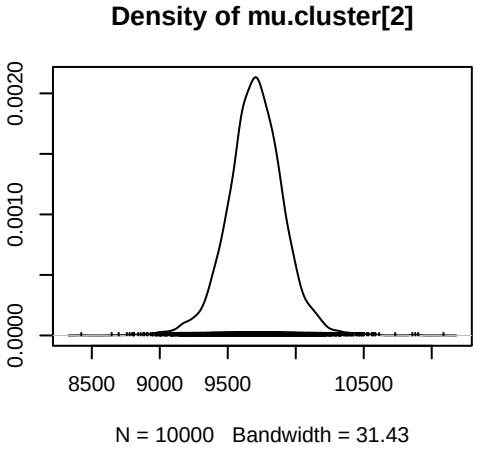
```
modelString <- textConnection("model{

  # Likelihood
  for(i in 1:n)
  {
    Y[i] ~ dnorm(mu[i], tau[i])
    mu[i] = mu.cluster[z[i]]
    tau[i] = tau.cluster[z[i]]
    z[i] ~ dcat(pi[1:K])
  }

  # Priors
  for(j in 1:K)
  {
    mu.cluster[j] ~ dnorm(0, 10^(-10))
    tau.cluster[j] ~ dgamma(0.01, 0.01)
    sig2.cluster[j] = 1/tau.cluster[j]
  }
  pi[1:K] ~ ddirch(c(1, 1, 1))
```

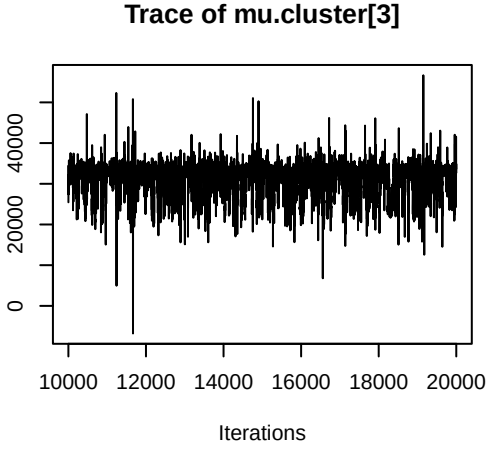
```
  for(g in 1:G)
  {
    y[g] = pi[1] * dnorm(y.grid[g], mu.cluster[1], tau.cluster[1]) +
    pi[2] * dnorm(y.grid[g], mu.cluster[2], tau.cluster[2]) +
    pi[3] * dnorm(y.grid[g], mu.cluster[3], tau.cluster[3])
  }

}")

# The required grid
y.grid <- seq(from = 5000, to = 40000, by = 100)
G <- length(y.grid)

data.list <- list(Y = Y,
                  n = length(Y),
                  K = 3,
                  y.grid = y.grid,
                  G = G)

model <- jags.model(file = modelString,
                    data = data.list,
                    n.chains = 1, quiet = T)
update(model, n.iter = 1e4)

# generating samples
params <- c("mu.cluster", "sig2.cluster", "pi", "y")
samples <- coda.samples(model = model,
                        variable.names = params,
                        n.iter = 1e4)

plot(samples[[1]][, 1:3])
```

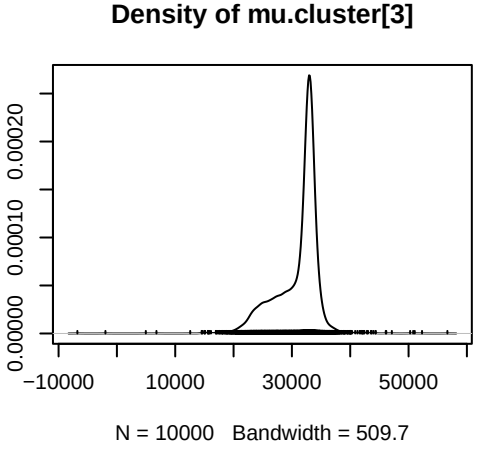




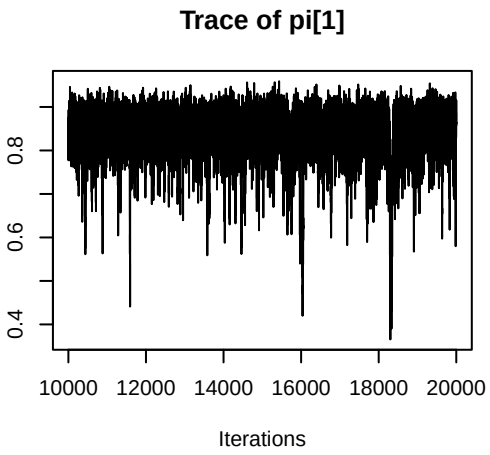


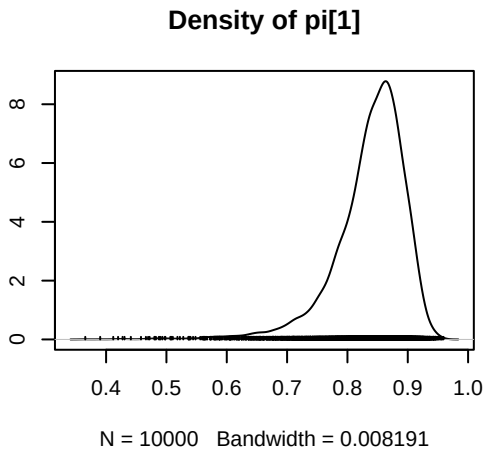


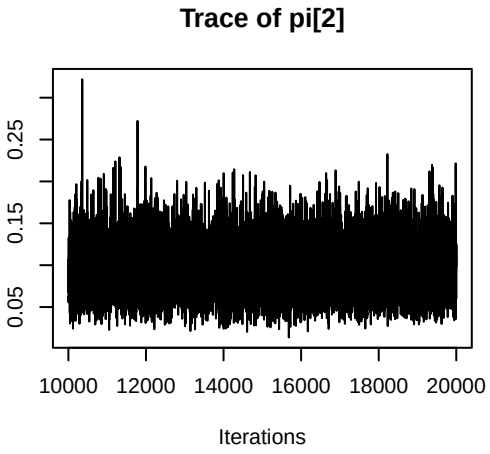


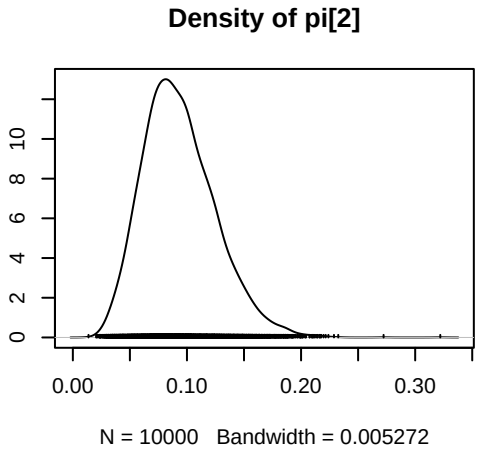


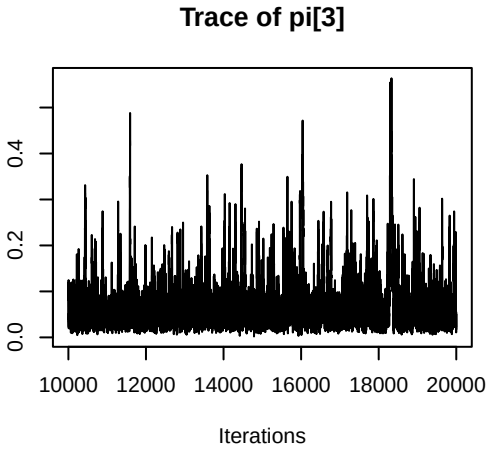

```
plot(samples[[1]][, 4:6])
```

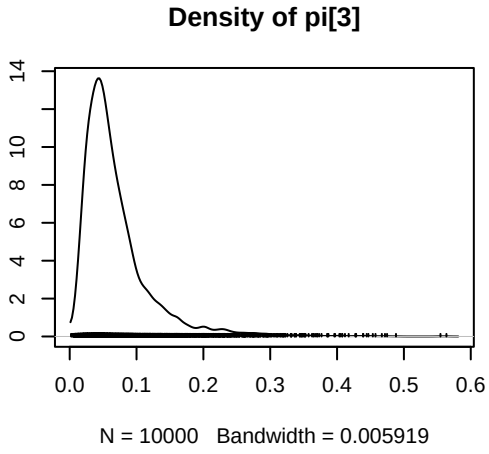




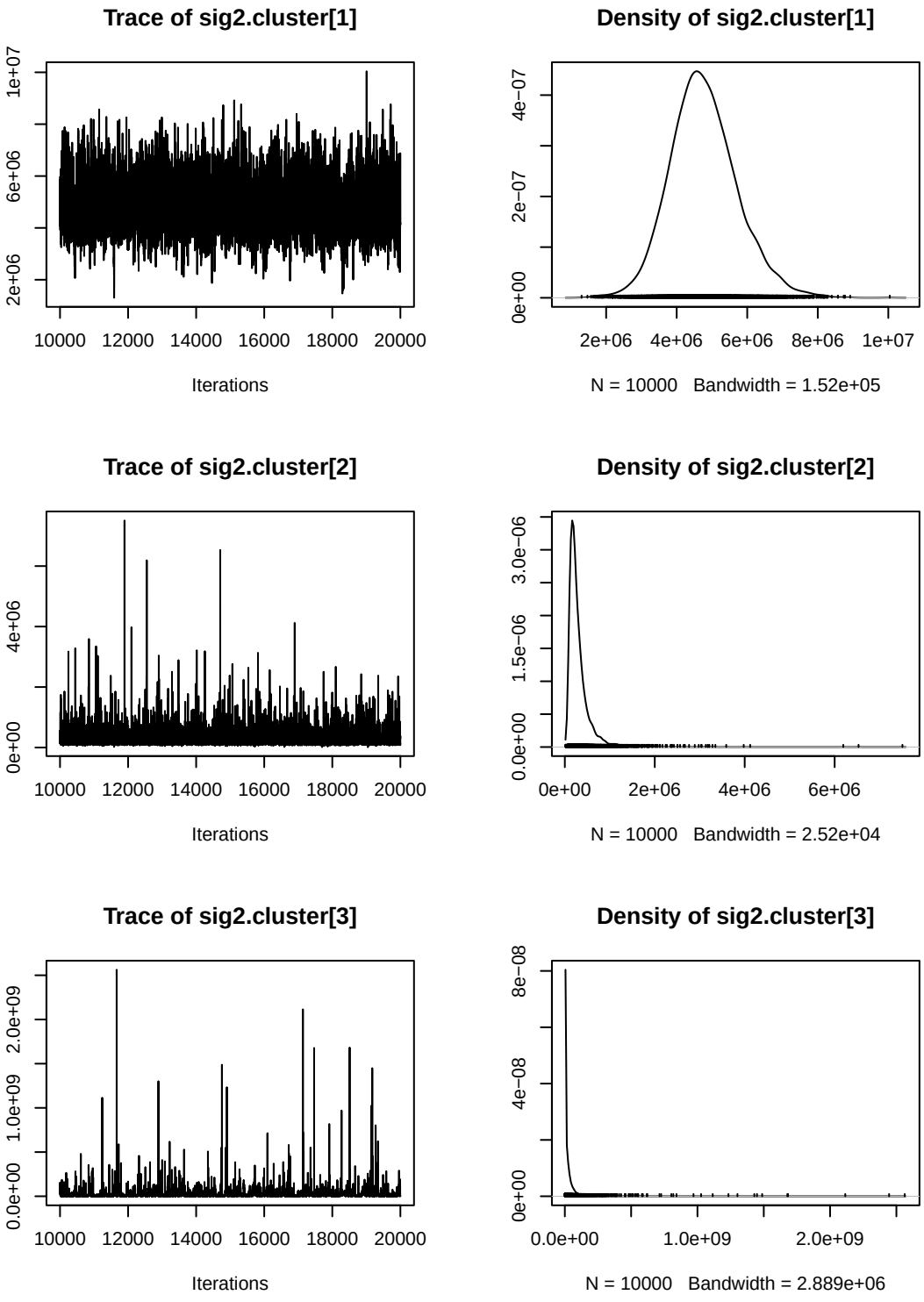








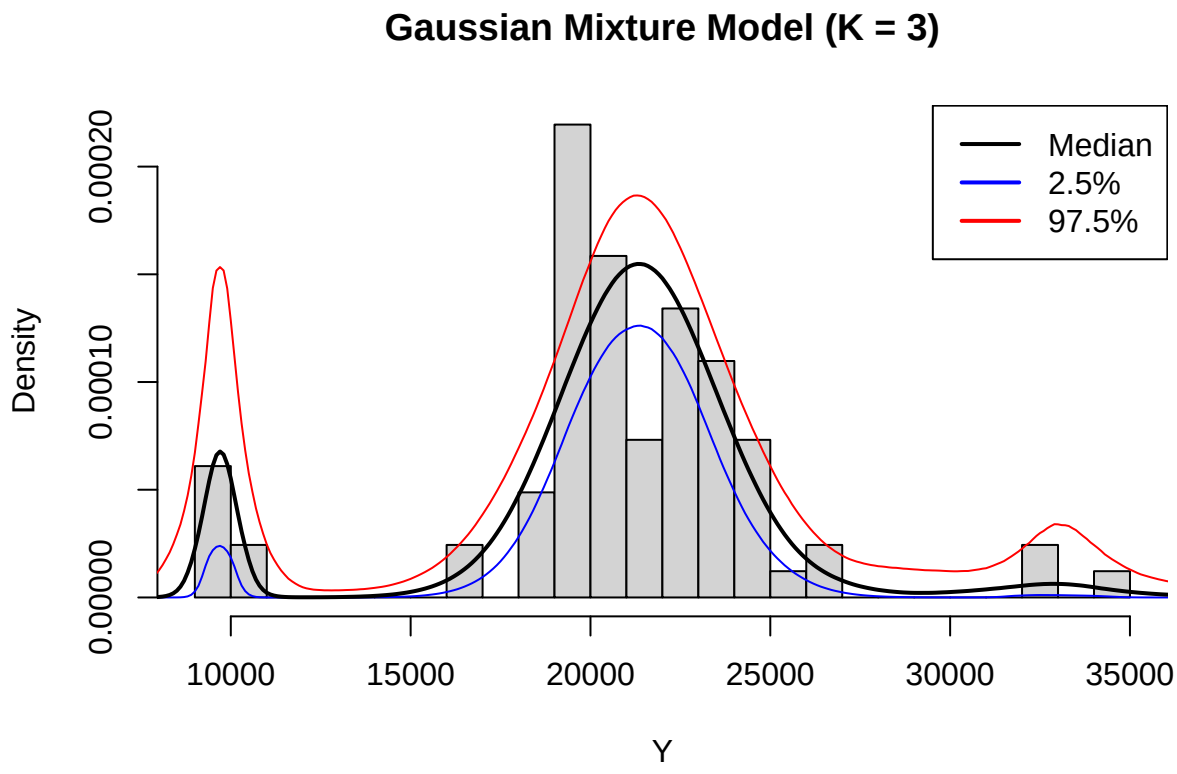

```
plot(samples[[1]][, 7:9])
```



From the trace plots we can observe that the MCMC chain has converged.

Posterior density plot

```
y.density.samples <- as.matrix(samples[[1]][ ,10:(G+10-1)])
y.density.median <- apply(X = y.density.samples,
                          MARGIN = 2, FUN = median)
y.density.lower <- apply(X = y.density.samples, MARGIN = 2,
                         FUN = function(x){quantile(x , p = 0.025)})
y.density.upper <- apply(X = y.density.samples, MARGIN = 2,
                         FUN = function(x){quantile(x , p = 0.975)})

hist(Y, probability = T, breaks = 25,
     xlab = "Y", ylab = 'Density',
     main = "Gaussian Mixture Model (K = 3)")
lines(x = y.grid, y = y.density.upper, type = 'l', col = 'red', lwd = 1)
lines(x = y.grid, y = y.density.lower, col = 'blue', lwd = 1)
lines(x = y.grid, y = y.density.median, col = 'black', lwd = 2)
legend("topright", legend = c("Median", "2.5%", "97.5%"),
       lty = 1, col = c("black", "blue", "red"), lwd = 2)
```



Conclusion

From the plot, we can observe that the model fits the data well.

Q3 (Ch: 5-4)

We are given the data: $Y_1 = 563, N_1 = 2820$ and $Y_2 = 10, N_2 = 27$. And we have to compare between two models:

$M_1 : Y_i|\lambda_i \sim Poisson(N_i\lambda_i)\ i = 1, 2$

$M_2 : Y_i|\lambda_0 \sim Poisson(N_i\lambda_0)\ i = 1, 2$

Bayes Factor

The given snippet is for c = 1. Similarly, we can evaluate BF for c = 10.

```
c <- 1

num.gamma1 <- pgamma(q = 1, shape = Y1+1, rate = N1)
num.gamma2 <- pgamma(q = 1, shape = Y2+1, rate = N2)
num.gamma <- num.gamma1 * num.gamma2

denom.gamma <- pgamma(q = 1, shape = Y1+Y2+1, rate = N1+N2)

const1 <- num.gamma / (denom.gamma * c * choose(Y1+Y2, Y1) * N2)
const2 <- exp((Y1+Y2+1)*log(1 + N2/N1) - (Y2)*log(N2/N1))

BF1.12 <- const1 * const2
```

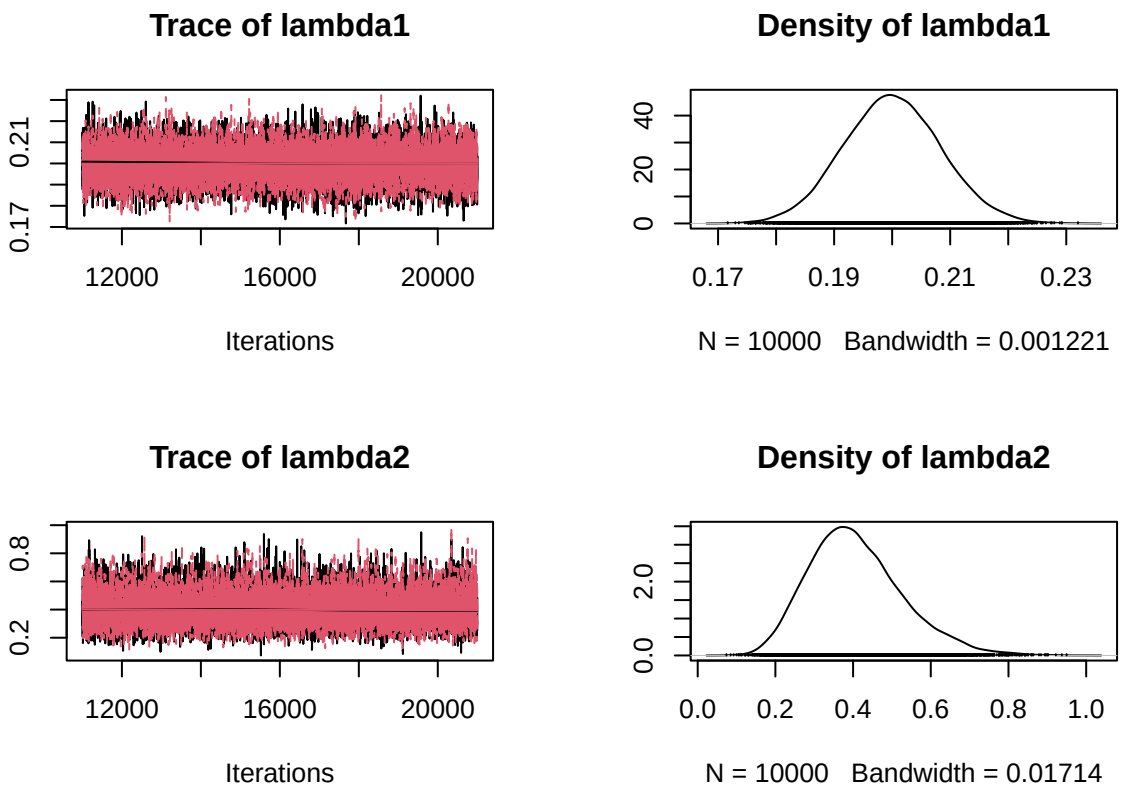
```
##        c = 1    c = 10
## BF  1.397494 0.1397494
```

DIC and WAIC (c = 1)

JAGS code

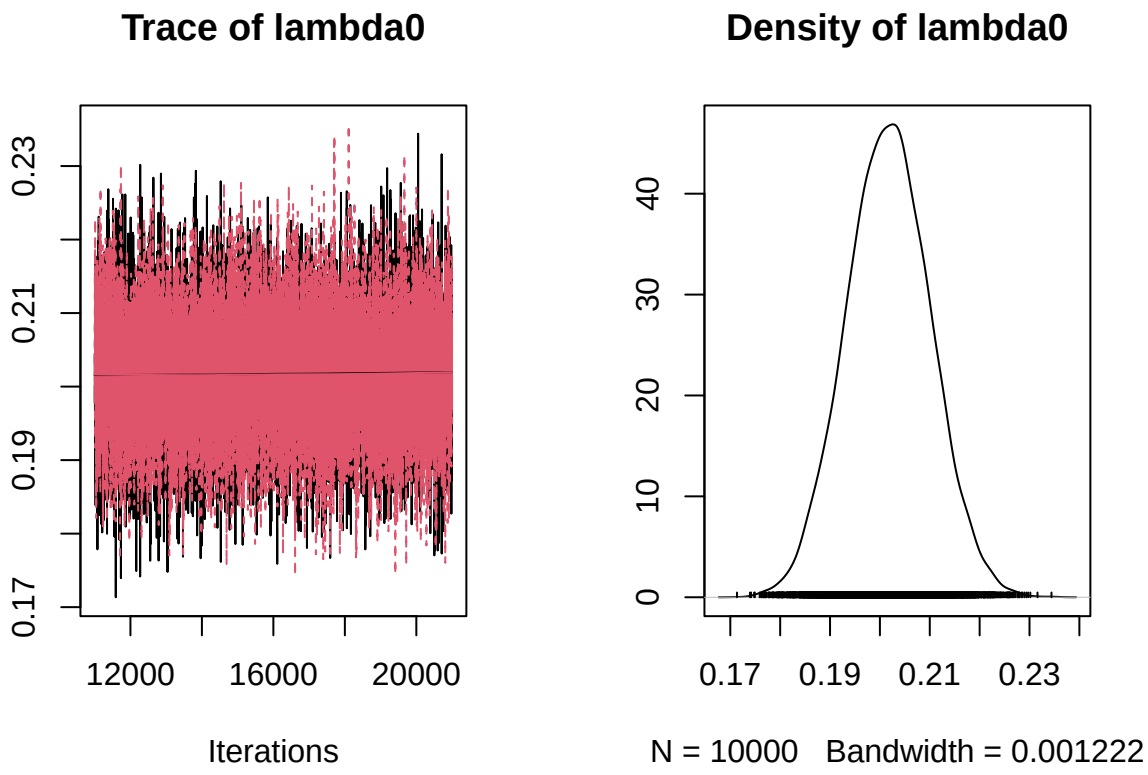
```
c <- 1

data.list <- list(Y1 = Y1, N1 = N1,
                  Y2 = Y2, N2 = N2,
                  c = c)

modelString1 <- textConnection("model{

  # Likelihood
  Y1 ~ dpois(N1 * lambda1)
  Y2 ~ dpois(N2 * lambda2)

  # Prior
  lambda1 ~ dunif(0, c)
  lambda2 ~ dunif(0, c)

}")
```

```
modelString2 <- textConnection("model{

  # Likelihood
  Y1 ~ dpois(N1 * lambda0)
  Y2 ~ dpois(N2 * lambda0)

  # Prior
  lambda0 ~ dunif(0, c)

}")

### Model 1
model1 <- jags.model(file = modelString1,
                     data = data.list,
                     n.chains = 2, quiet = T)
update(model1, n.iter = 1e4)
params <- c("lambda1", "lambda2")
samples1 <- coda.samples(model = model1,
                         n.iter = 1e4,
                         variable.names = params)

plot(samples1)
```




```
lambda1 <- samples1[[1]][ ,1]
lambda2 <- samples1[[1]][ ,2]
```

```
### Model 2
model2 <- jags.model(file = modelString2,
                     data = data.list,
                     n.chains = 2, quiet = T)
update(model2, n.iter = 1e4)
params <- c("lambda0")
samples2 <- coda.samples(model = model2,
                         n.iter = 1e4,
                         variable.names = params)

plot(samples2)
```




```
lambda0 <- samples2[[1]][ ,1]
```

DIC

```
loglike.m1.func <- function(iter)
{
  iter.lambda1 <- lambda1[iter]
  iter.lambda2 <- lambda2[iter]

  loglike.Y1 <- dpois(x = Y1, N1 * iter.lambda1, log = T)
```

```
  loglike.Y2 <- dpois(x = Y2, N2 * iter.lambda2, log = T)

  return(loglike.Y1 + loglike.Y2)
}

loglike.m2.func <- function(iter)
{
  iter.lambda <- lambda0[iter]

  loglike.Y1 <- dpois(x = Y1, N1 * iter.lambda, log = T)
  loglike.Y2 <- dpois(x = Y2, N2 * iter.lambda, log = T)

  return(loglike.Y1 + loglike.Y2)
}

loglike.m1 <- sapply(1:1e4, FUN = loglike.m1.func)
loglike.m2 <- sapply(1:1e4, FUN = loglike.m2.func)

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

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

loglike.thetahat.m1 <- dpois(Y1, lambda = N1 * mean(lambda1), log = T) +
                       dpois(Y2, lambda = N2 * mean(lambda2), log = T)
D.thetahat.m1 <- -2 * loglike.thetahat.m1

loglike.thetahat.m2 <- dpois(Y1, lambda = N1 * mean(lambda0), log = T) +
                       dpois(Y2, lambda = N2 * mean(lambda0), log = T)
D.thetahat.m2 <- -2 * loglike.thetahat.m2

pD.m1 <- Dbar.m1 - D.thetahat.m1
pD.m2 <- Dbar.m2 - D.thetahat.m2

DIC.m1 <- pD.m1 + Dbar.m1
DIC.m2 <- pD.m2 + Dbar.m2

DIC.c1 <- c(DIC.m1, DIC.m2)

paste("DIC (c = 1): ", DIC.c1)
```

```
## [1] "DIC (c = 1):  16.2395326128794" "DIC (c = 1):  17.4083257505227"
```

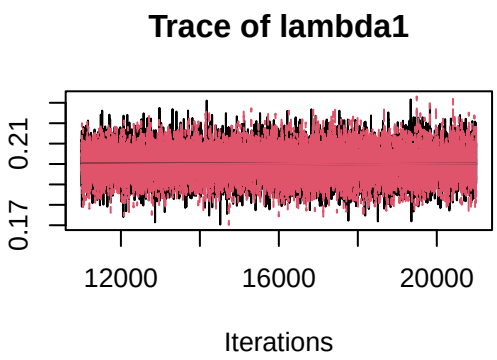
WAIC

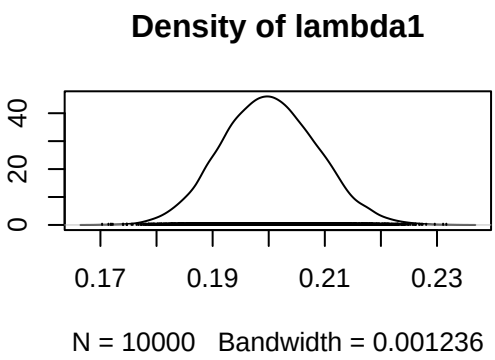
```
loglike.m1.func <- function(iter)
{
  iter.lambda1 <- lambda1[iter]
  iter.lambda2 <- lambda2[iter]
```

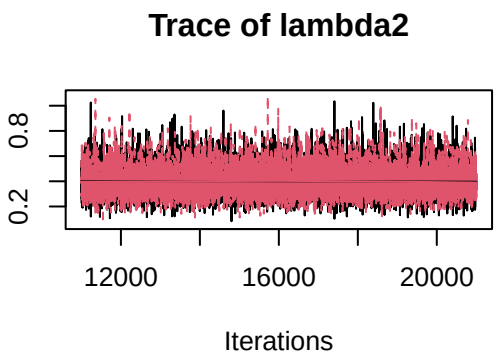
```
  loglike.Y1 <- dpois(x = Y1, N1 * iter.lambda1, log = T)
  loglike.Y2 <- dpois(x = Y2, N2 * iter.lambda2, log = T)

  return(c(loglike.Y1, loglike.Y2))
}

loglike.m2.func <- function(iter)
{
  iter.lambda <- lambda0[iter]

  loglike.Y1 <- dpois(x = Y1, N1 * iter.lambda, log = T)
  loglike.Y2 <- dpois(x = Y2, N2 * iter.lambda, log = T)

  return(c(loglike.Y1, loglike.Y2))
}

loglike.m1 <- sapply(1:1e4, FUN = loglike.m1.func)
loglike.m2 <- sapply(1:1e4, FUN = loglike.m2.func)

posmeans.m1 <- apply(loglike.m1, 1, mean)
posmeans.m2 <- apply(loglike.m2, 1, mean)

posvars.m1 <- apply(loglike.m1, 1, var)
posvars.m2 <- apply(loglike.m2, 1, var)

pW.m1 <- sum(posvars.m1)
pW.m2 <- sum(posvars.m2)

sum.means.m1 <- sum(posmeans.m1)
sum.means.m2 <- sum(posmeans.m2)

WAIC.m1 <- -2 * sum.means.m1 + 2 * pW.m1
WAIC.m2 <- -2 * sum.means.m2 + 2 * pW.m2

WAIC.c1 <- c(WAIC.m1, WAIC.m2)
paste("WAIC (c = 1): ", WAIC.c1)
```

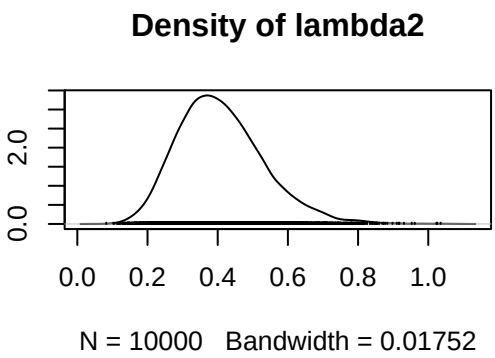
```
## [1] "WAIC (c = 1):  16.3190558619896" "WAIC (c = 1):  17.5161000937042"
```

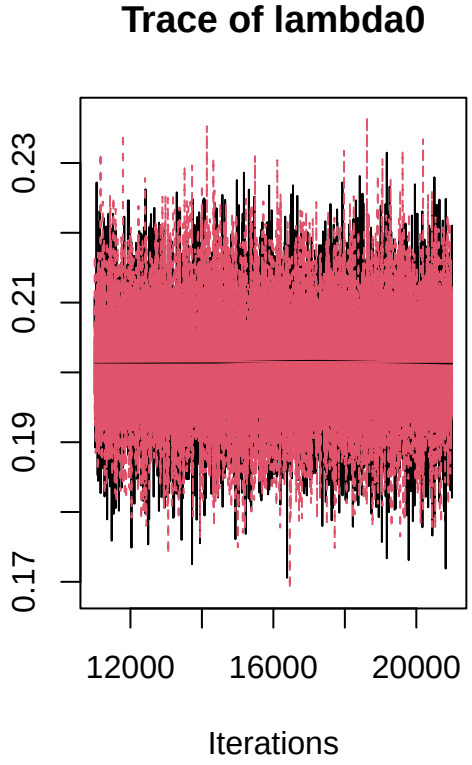
DIC and WAIC (c = 10)

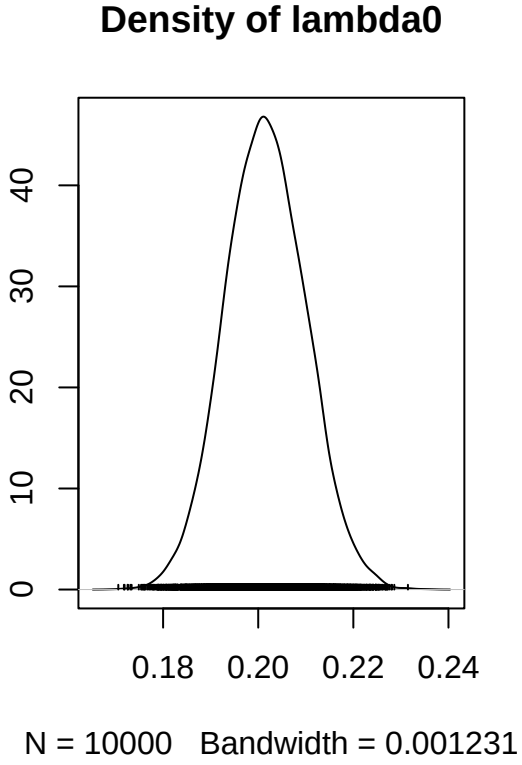












```
## [1] "DIC (c = 10):  16.2347466368283" "DIC (c = 10):  17.4075514019931"
```

```
## [1] "WAIC (c = 10):  16.3446638564767" "WAIC (c = 10):  17.4998354637415"
```

Conclusion

```
##        c = 1    c = 10
## BF  1.397494 0.1397494
```

The BF for model M_1 compared to model M_2 is < 10. Hence we cannot conclude which model is better using BF.

However, if we have to choose a model, we can prefer M1 as it still gives lower DIC and WAIC compared to M2.

```
##                      M1       M2
## DIC (c = 1)   16.23953 17.40833
## DIC (c = 10)  16.23475 17.40755
## WAIC (c = 1)  16.31906 17.51610
## WAIC (c = 10) 16.34466 17.49984
```

In all cases, DIC and WAIC is smaller for M_1 compared to M_2. However, there is not any substantial evidence to prefer any one model.

Q4 (Ch: 5-6)

We have to fit logistic regression model without any random effects to the gambia dataset, and use PPD checks to check if the model fits well. Since Y is either 0 or 1, using min, max, or median will not give meaningful results. Hence I have used **mean** and **SD** for PPD checks.

```
library(geoR)
library(rjags)

### Preparing the data
data("gambia")
Y <- gambia$pos

# Added column of 1's for beta_0
X <- as.matrix(subset(x = gambia, select = -c(x, y, pos)))
X <- scale(X)
X <- cbind(1, X)

print(head(gambia))
```

```
##             x       y pos  age netuse treated green phc
## 1850 349631.3 1458055   1 1783      0       0 40.85   1
## 1851 349631.3 1458055   0  404      1       0 40.85   1
## 1852 349631.3 1458055   0  452      1       0 40.85   1
## 1853 349631.3 1458055   1  566      1       0 40.85   1
## 1854 349631.3 1458055   0  598      1       0 40.85   1
## 1855 349631.3 1458055   1  590      1       0 40.85   1
```

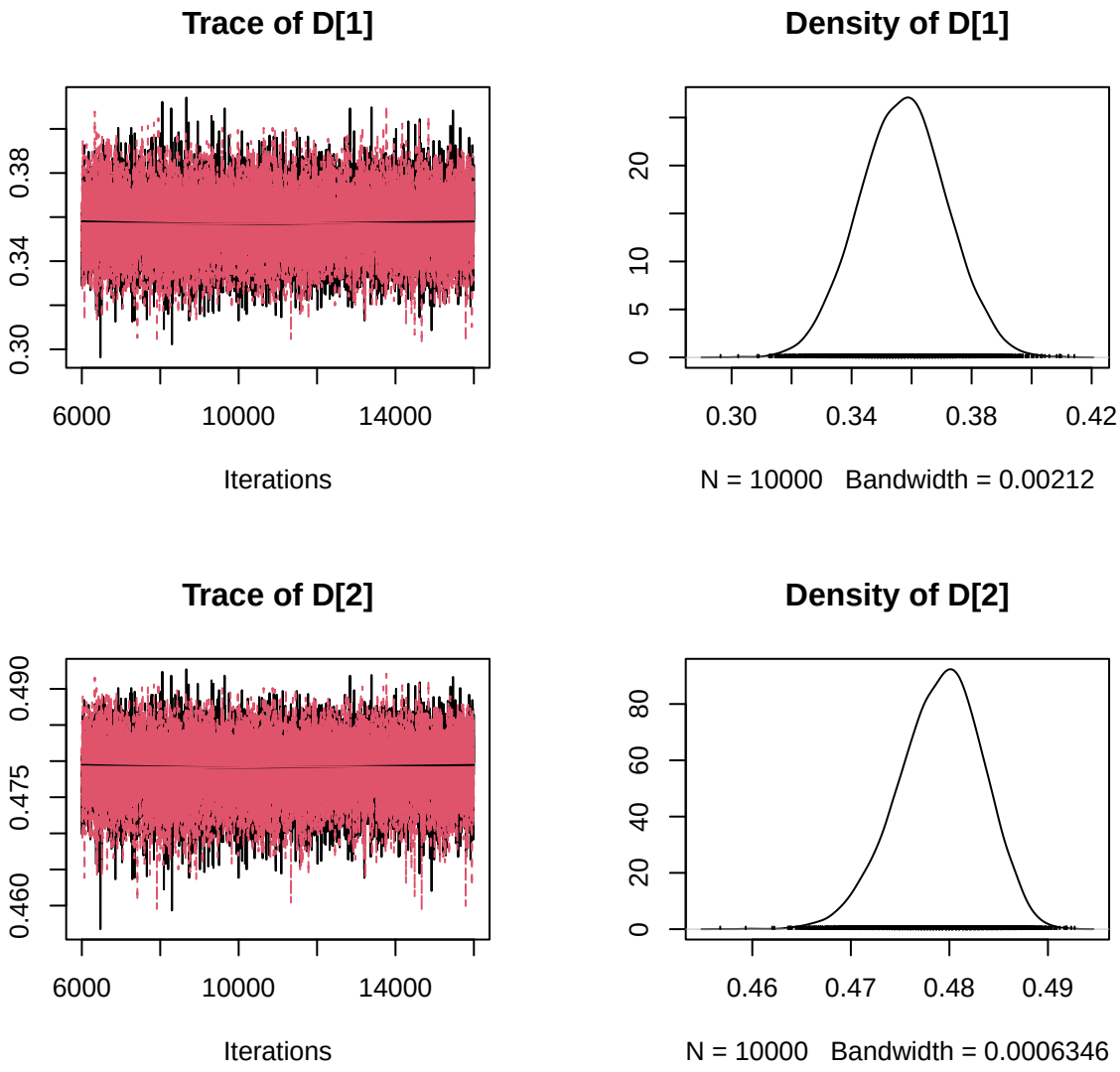
JAGS code

Note: Since it was taking a long time to run the MCMC chain each time, I have saved the samples from earlier run in an .Rdata file and I am using them here.

```
### JAGS model
modelString <- textConnection("model{

  # Likelihood
  for(i in 1:n)
  {
    Y[i] ~ dbern(q[i])
    logit(q[i]) = inprod(X[i, ], beta[])
  }

  # Priors
  for(j in 1:p)
  {
    beta[j] ~ dnorm(0, 100^(-2))
  }

  # PPD
  for(i in 1:n)
  {
    Yppd[i] ~ dbern(t[i])
    logit(t[i]) = inprod(X[i, ], beta[])
  }

  D[1] <- mean(Yppd[])
  D[2] <- sd(Yppd[])

}")

data.list <- list(Y = Y, X = X,
                  n = length(Y),
                  p = length(X[1, ]))
model <- jags.model(file = modelString,
                    data = data.list,
                    n.chains = 2)

# Burn-in
update(model, 5e3)

# Samples

n.samples <- 1e4
params <- c("D")

samples <- coda.samples(model = model,
                        variable.names = params,
                        n.iter = n.samples)

# out <- list(model = model,
#             samples = samples)
```

```
# save(out, file = 'out-6.Rdata')
```

```
load('./out-6.Rdata')
model <- out$model
samples <- out$samples
plot(samples)
```

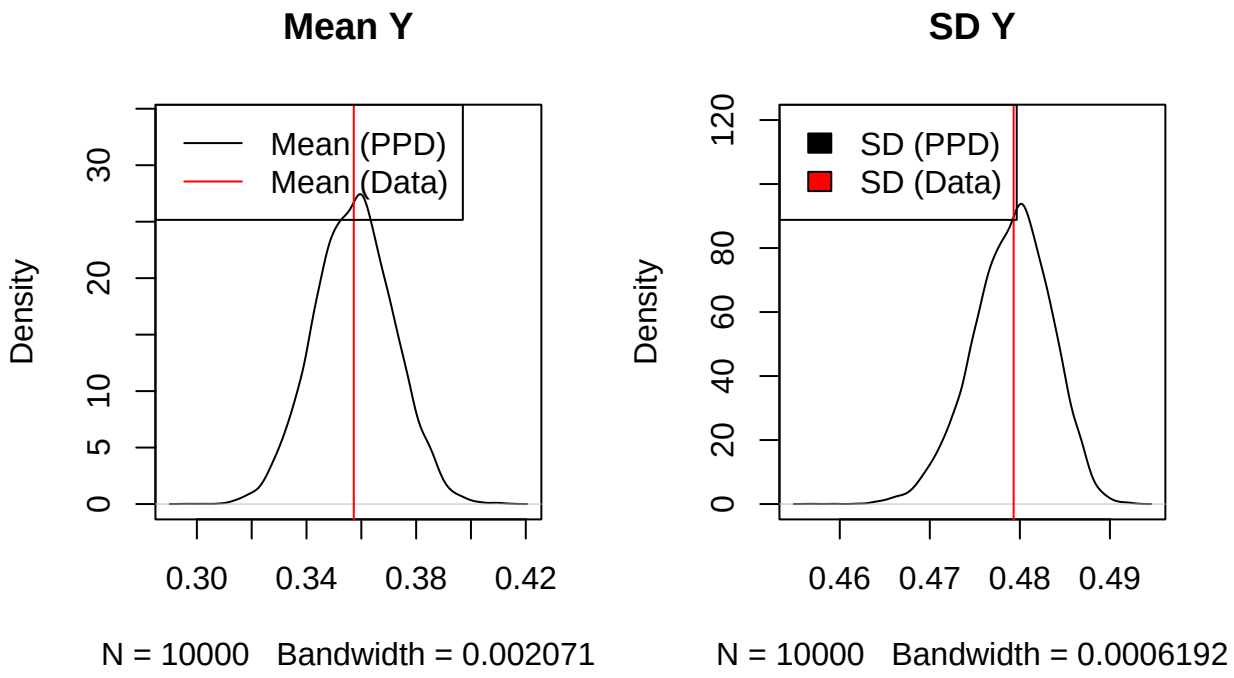


The MCMC chains have converged for both Mean (D[1]) and SD (D[2])

PPD checks

```
ppd.mean <- samples[[1]][ ,1]
ppd.sd <- samples[[1]][ ,2]

par(mfrow = c(1, 2))
```

```r
plot(density(ppd.mean), main = "Mean Y", ylim = c(0, 34))
abline(v = mean(Y), col = 'red')
legend("topleft", legend = c("Mean (PPD)", "Mean (Data)"),
       lty = 1, col = c('black', 'red'))
pval1 <- mean(ppd.mean > mean(Y))

plot(density(ppd.sd), main = "SD Y", ylim = c(0, 120))
abline(v = sd(Y), col = 'red')
legend("topleft", legend = c("SD (PPD)", "SD (Data)"), fill = c('black', 'red'))
```




```r
pval2 <- mean(ppd.sd > sd(Y))

p.table <- matrix(data = c(pval1, pval2), nrow = 1)
colnames(p.table) <- c("Mean", "SD")

print(p.table)
```

```
##        Mean     SD
## [1,] 0.4986 0.5092
```

Conclusion

From the plots and p value, we can conclude that the model fits the data well.

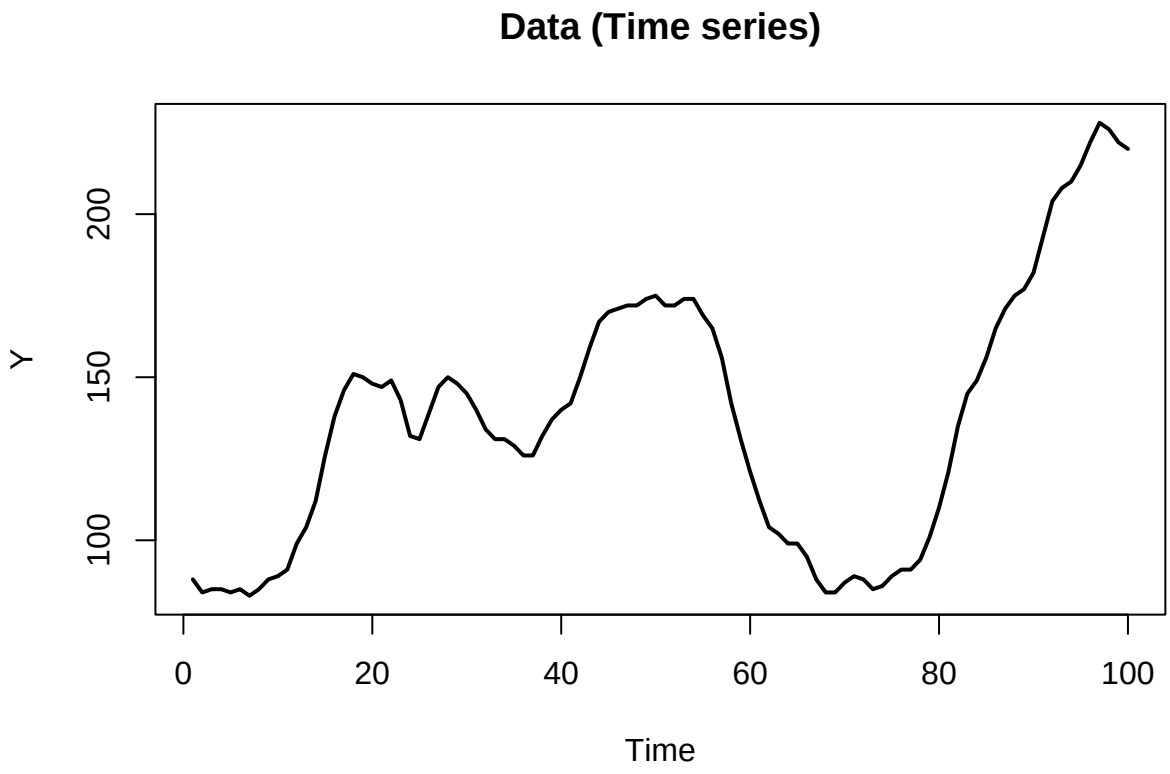
Q5 (Ch: 5-10)

We are a time series dataset where Y_t is the WWW usage at time t. For L = 1, 2, 3, 4, we have to fit the following AR model:

$$Y_t|Y_{t-1}, ..., Y_1 \sim Normal(\beta_0 + \beta_1 Y_{t-1} + ... + \beta_L Y_{t-L}, \sigma^2)$$

And among the various models, we have to choose the best one. To choose the best model, I will use the DIC criteria.

```
library(datasets)
library(rjags)

data("WWWusage")
Y <- as.numeric(WWWusage)

plot(WWWusage, type = 'l', lwd = 2,
     ylab = "Y", main = "Data (Time series)")
```



JAGS code

```
# JAGS model
modelString <- "model{
```

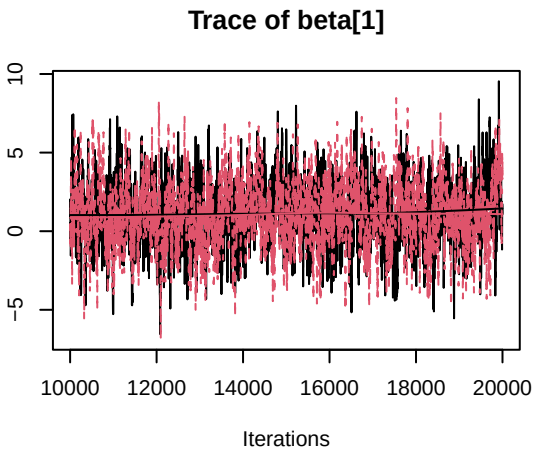
```
  # Likelihood
  for(i in 5:n)
  {
    Y[i] ~ dnorm(mu[i], tau)
    mu[i] <- beta[1] + sum(Y[(i-L):(i-1)] * beta[2:(L+1)])
  }

  # Prior
  for(j in 1:(L+1))
  {
    beta[j] ~ dnorm(0, 100^(-2))
  }
  tau ~ dgamma(0.01, 0.01)
  sig2 = 1/tau

}"
```

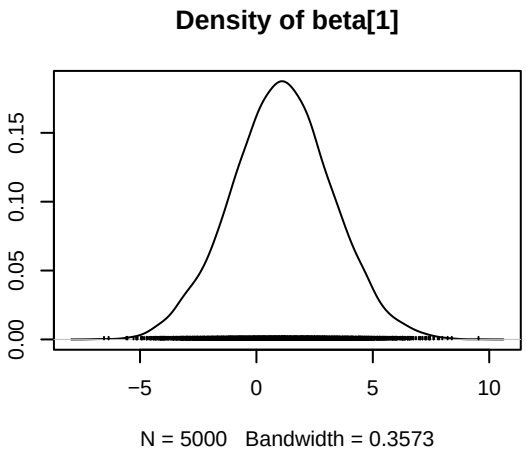
Different values of L

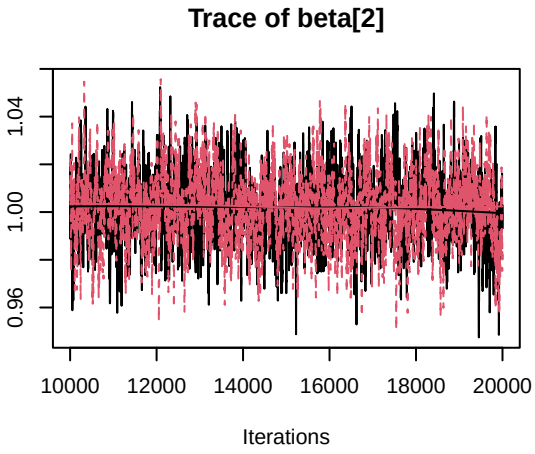
I have given below the code for (L = 1) case. For other values of L = 2, 3, 4, I have only shown the plots of MCMC samples. Finally I compare the DIC for different models to choose the best one.

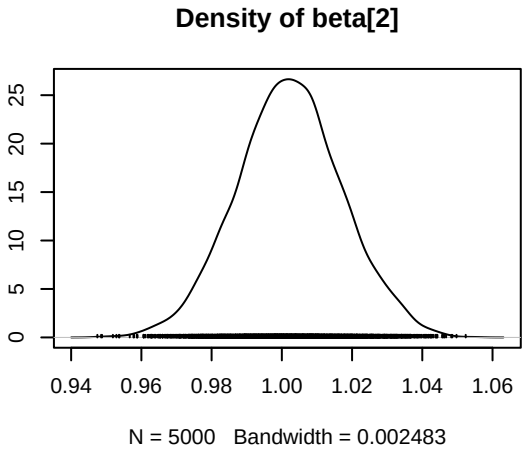
L = 1

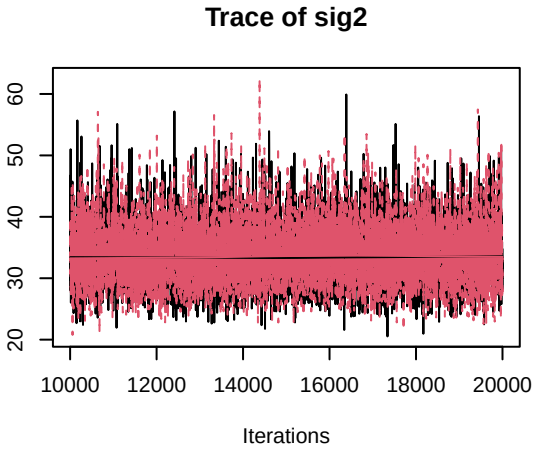
```
# L = 1
data.list1 <- list(Y = Y, n = length(Y), L = 1)
model1 <- jags.model(file = textConnection(modelString),
                     data = data.list1,
                     n.chains = 2, quiet = T)
update(model1, n.iter = 1e4)
params <- c("beta", "sig2")
samples1 <- coda.samples(model = model1,
                         variable.names = params,
                         n.iter = 1e4,
                         thin = 2)
plot(samples1)
```

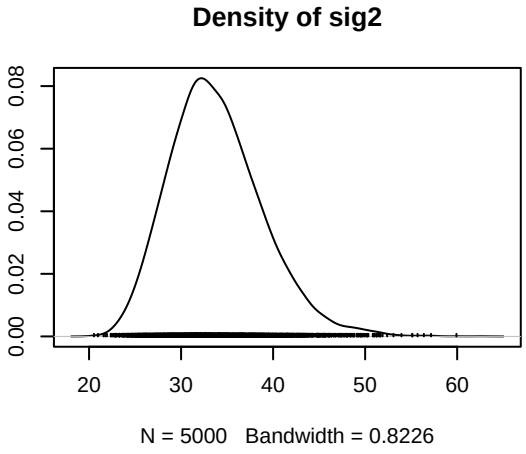






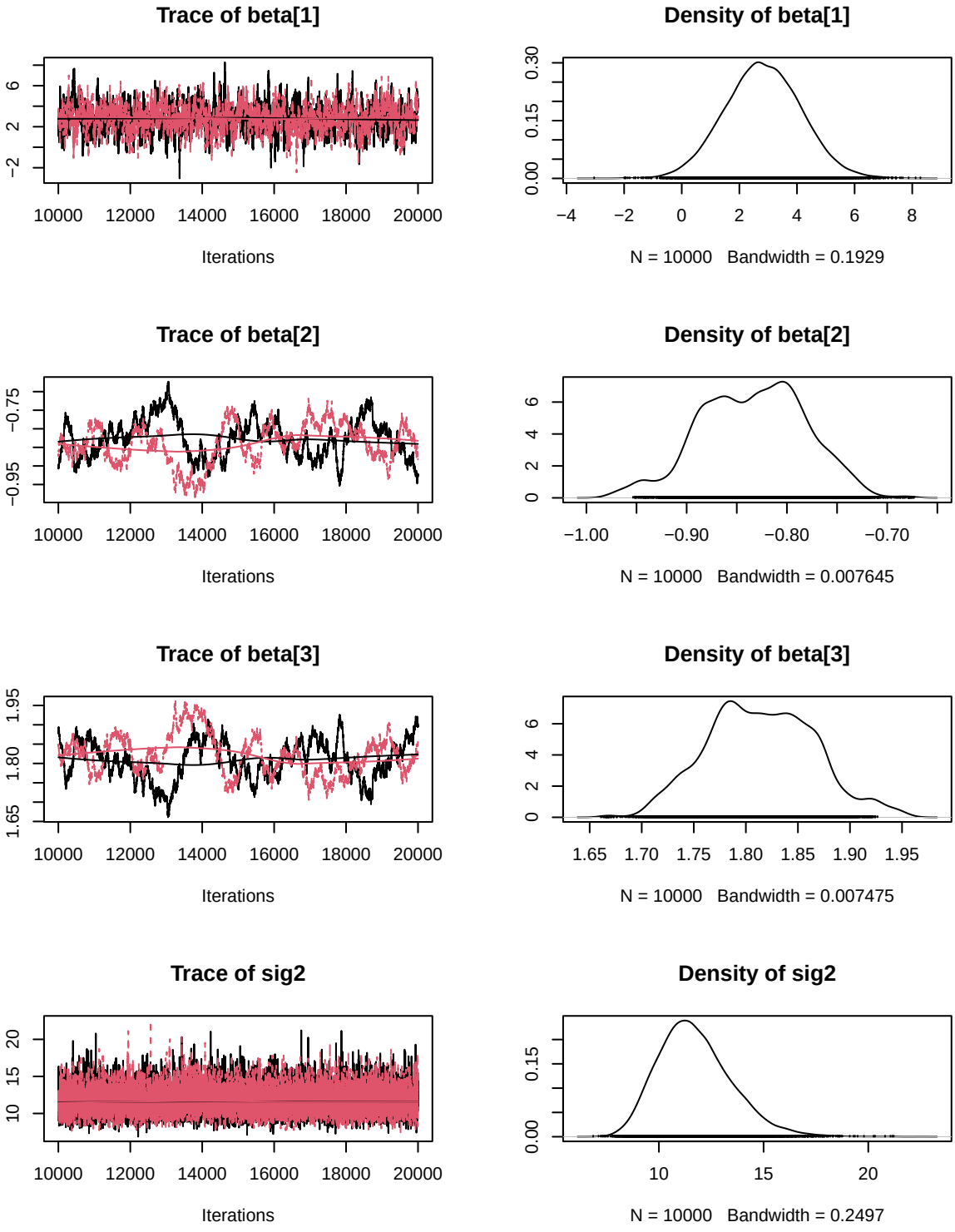




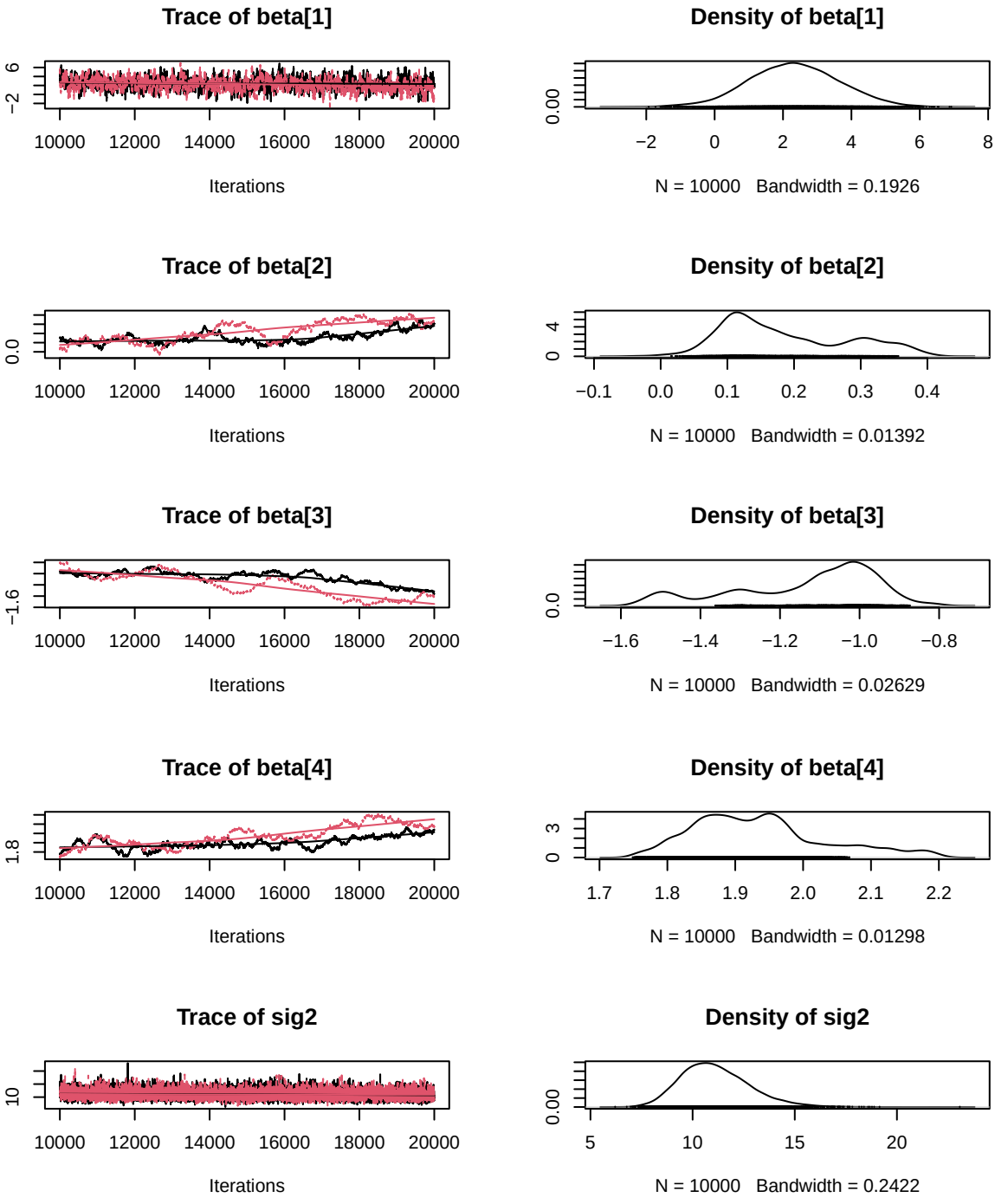



```
dic1 <- dic.samples(model = model1, n.iter = 1e4)
```

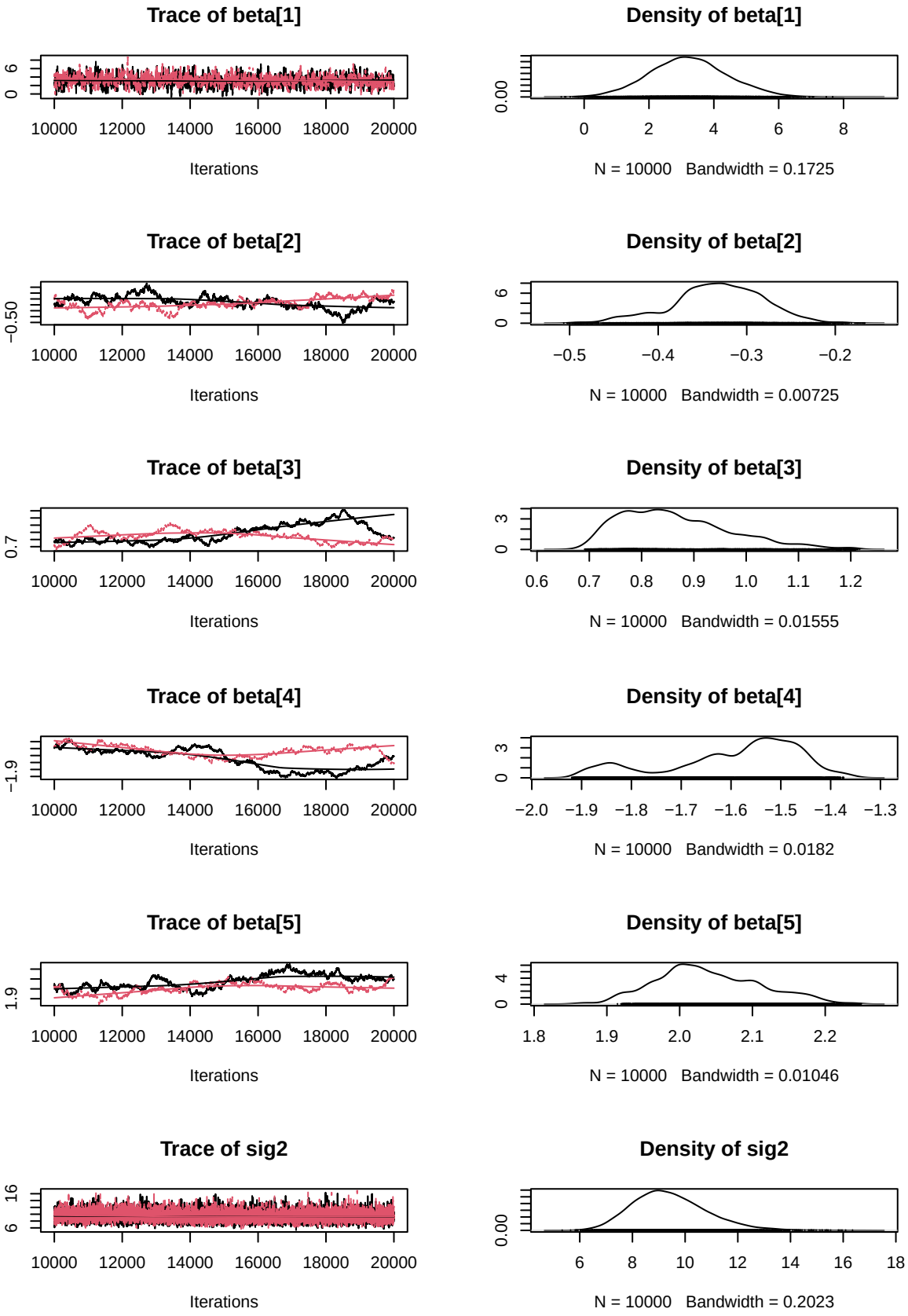
L = 2


Trace of beta[1]
Iterations
Density of beta[1]
N = 10000 Bandwidth = 0.1929
Trace of beta[2]
Iterations
Density of beta[2]
N = 10000 Bandwidth = 0.007645
Trace of beta[3]
Iterations
Density of beta[3]
N = 10000 Bandwidth = 0.007475
Trace of sig2
Iterations
Density of sig2
N = 10000 Bandwidth = 0.2497

L = 3


Trace of beta[1]
Density of beta[1]
Iterations
N = 10000 Bandwidth = 0.1926
Trace of beta[2]
Density of beta[2]
Iterations
N = 10000 Bandwidth = 0.01392
Trace of beta[3]
Density of beta[3]
Iterations
N = 10000 Bandwidth = 0.02629
Trace of beta[4]
Density of beta[4]
Iterations
N = 10000 Bandwidth = 0.01298
Trace of sig2
Density of sig2
Iterations
N = 10000 Bandwidth = 0.2422

$\mathbf{L = 4}$


Trace of beta[1]
Density of beta[1]
Iterations
N = 10000 Bandwidth = 0.1725
Trace of beta[2]
Density of beta[2]
Iterations
N = 10000 Bandwidth = 0.00725
Trace of beta[3]
Density of beta[3]
Iterations
N = 10000 Bandwidth = 0.01555
Trace of beta[4]
Density of beta[4]
Iterations
N = 10000 Bandwidth = 0.0182
Trace of beta[5]
Density of beta[5]
Iterations
N = 10000 Bandwidth = 0.01046
Trace of sig2
Density of sig2
Iterations
N = 10000 Bandwidth = 0.2023

Conclusion

The trace plots does not converge for β_i's for L = 2, 3, 4. However, from the DIC values are given below, we can see that we get lowest DIC for L = 4. ($DIC_1 > DIC_2 > DIC_3 > DIC_4$)

Since, the chains are diverging for L = 2, 3, 4, we cannot comment on the posterior distribution. Hence, I will prefer AR(1) process.

```
## [1] "L = 1"
```

```
## Mean deviance:  609.7
## penalty 3.055
## Penalized deviance: 612.8
```

```
## [1] "L = 2"
```

```
## Mean deviance:  508.3
## penalty 4.29
## Penalized deviance: 512.6
```

```
## [1] "L = 3"
```

```
## Mean deviance:  501.6
## penalty 4.168
## Penalized deviance: 505.7
```

```
## [1] "L = 4"
```

```
## Mean deviance:  486.5
## penalty 4.674
## Penalized deviance: 491.1
```