Lab 3

2024-05-02

```
library(BayesLogit)
library(mvtnorm)
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.32.6 (Stan version 2.32.2)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```
library(readxl) #to read excel files
```

#Question-1

#a) Consider again the logistic regression model in problem 2 from the previous computer lab 2. Use the prior beta N(0,tau^2 * I), where tau = 3.

Implement a Gibbs sampler that simulates from the joint posterior p(w, beta|x) by augmenting the data with Polya-gamma latent variables wi , i = 1, . . . , n. Evaluate the convergence of the Gibbs sampler by calculating the Inefficiency Factors (IFs) and by plotting the trajectories of the sampled Markov chains.

Given w = (w1, ..., wn), the conditional posterior of beta with prior beta N (b, B) follows a multivariate normal distribution.

$$\omega i|\beta \sim PG(1, xi'\beta), i = 1, ..., n.$$

PG() => solved using the rpg() function from the bayeslogit package.

$$\beta|y, \omega \sim N(m\omega, V\omega)$$

where

$$V\omega = (X^T * \Omega * X + B^-1)^-1$$

$$mw = Vw * (X^T\kappa + B^-1 * b)$$

where

$$\kappa = (y1 - 1/2, ..., yn - 1/2)$$

and Ω is the diagonal matrix of wi's.

Inefficiency factor:

$$1 + 2 \sum \rho k.$$

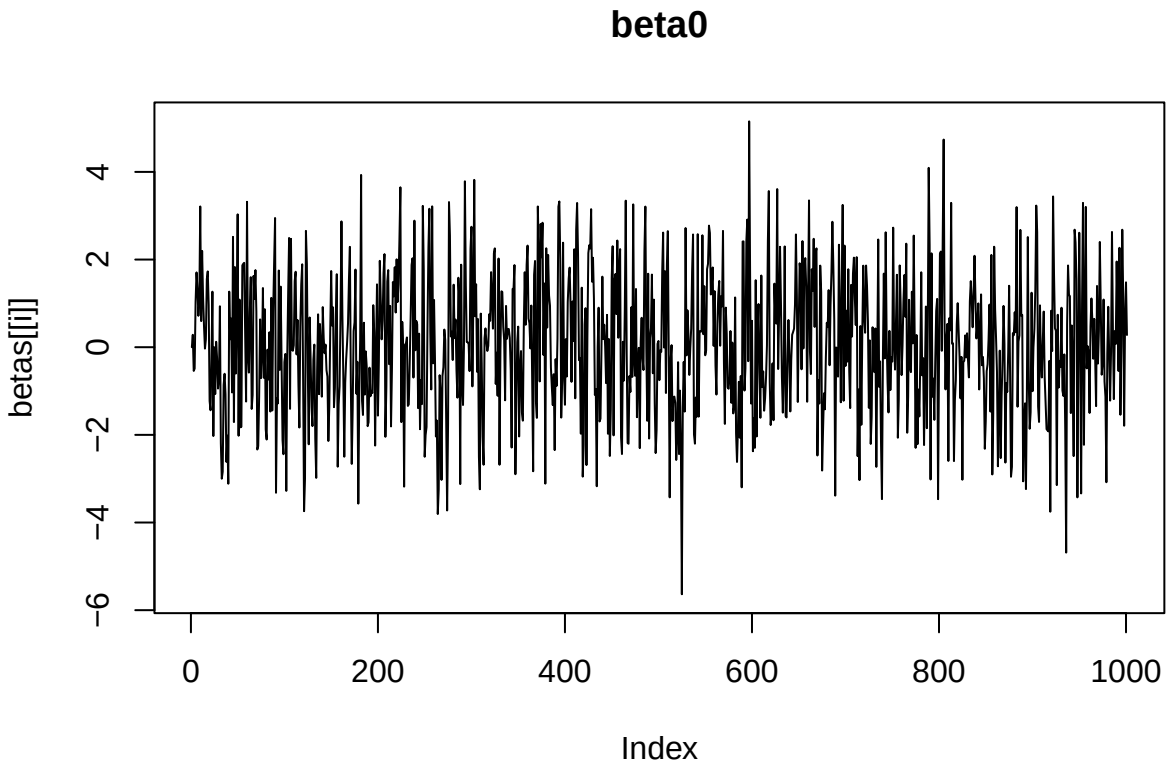
where pk is solved using the acf().

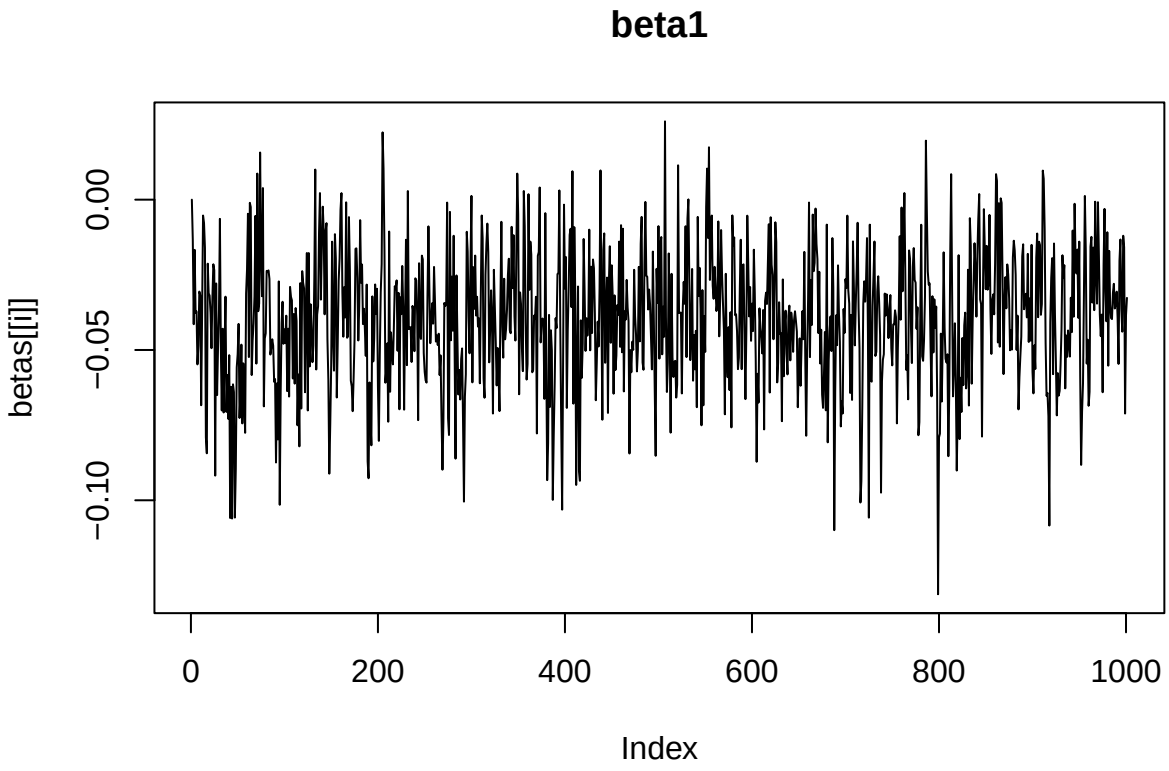
```
#Q.1) a)
women_data <- as.matrix(read.table("WomenAtWork.dat", header=TRUE))
n_params <- ncol(women_data) - 1
n_obs <- nrow(women_data)

y <- women_data[1:n_obs, 1]
X <- women_data[1:n_obs, 2:ncol(women_data)]
x_names <- colnames(X)

# Set up prior
tau <- 3
mu <- matrix(0, n_params) # Prior mean vector
Sigma <- tau ** 2 * diag(n_params)

# log_post_logistic <- function(betas, y, X, mu, Sigma) {
#   lin_pred <- X %*% betas
#   log_lik <- sum(lin_pred * y - log(1 + exp(lin_pred)))
#   log_prior <- dmvnorm(betas, mu, Sigma, log=TRUE)
#
#   return(log_lik + log_prior)
# }

# Initial values for beta
init_val <- matrix(0, n_params)

# Optimize betas
# optim_res <- optim(init_val, log_post_logistic, gr=NULL, y, X, mu, Sigma,
#                    method=c("BFGS"), control=list(fnscale=-1), hessian=TRUE)
# rownames(optim_res$par) <- x_names



###

beta<-list()
beta[[1]]<-init_val

draws<- 1000

j<-2

for (i in 1:draws){
  value1<-X %*% beta[[i]]

  w<-c()
  for (l in 1:n_obs){
    w[l]<-rpg(1,h=1, z =value1[l])
  }
```

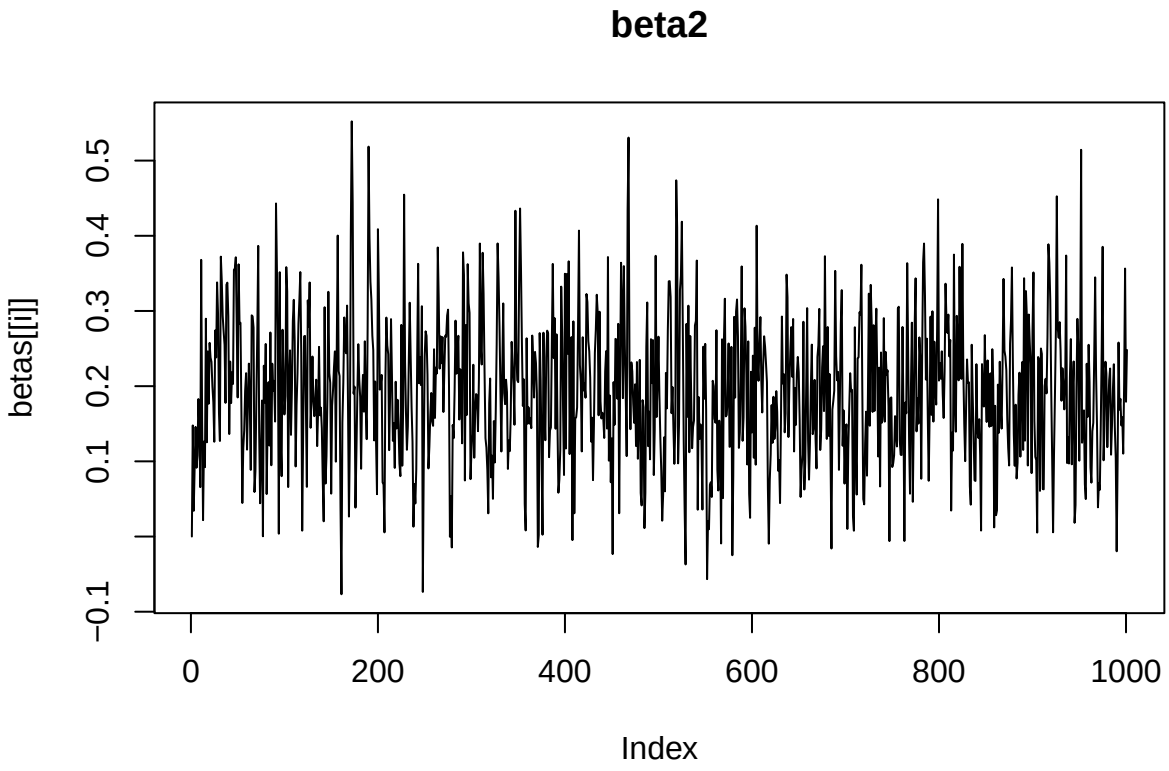
```
  Vw<-solve(t(X)%*%diag(w)%*%X+solve(Sigma))

  k<-y-0.5

  Mw<-Vw%*%(t(X)%*%k)

  beta[[j]]<-t(rmvnorm(1,mean = Mw,sigma = Vw))

  j<-j+1
}

beta_post<-c()

for (j in 1:7){
  values<-c()
  for (i in 1:length(beta)){
    values[i]<-beta[[i]][j]
  }
  beta_post[j]<-mean(values)
}

#beta_post #Values are similar to that of lab-2


# Calculating the IF for each of the coefficients using acf:
betas<-list()
for (j in 1:7){
  values<-c()
  for (i in 1:length(beta)){
    values[i]<-beta[[i]][j]
  }
  betas[[j]]<-values
}

I_factor<-c()
for (i in 1:7){
  a<-acf(betas[[i]],plot = FALSE)
  I_factor[i]<-1+2*sum(a$acf)
}

cat("IF values for all the betas:",I_factor) #For all the beta values
```

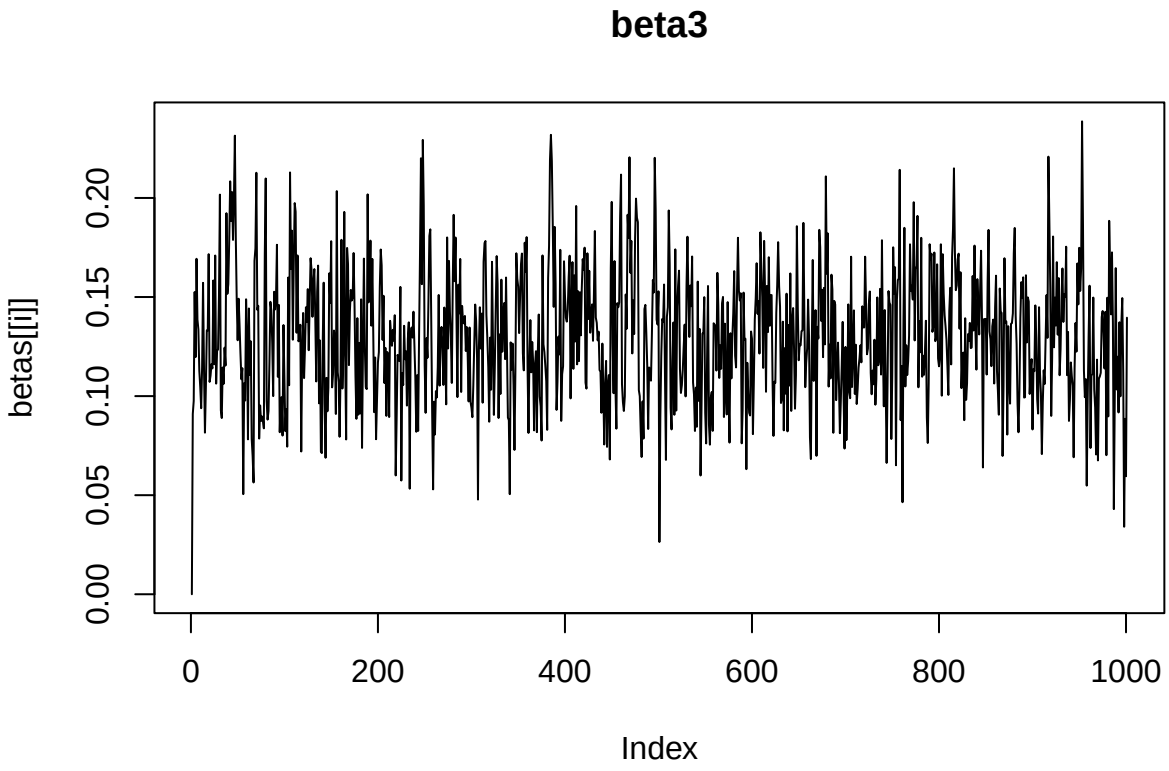
```
## IF values for all the betas: 2.988025 4.428518 3.327572 3.342465 3.469137 4.119445 3.991755
```

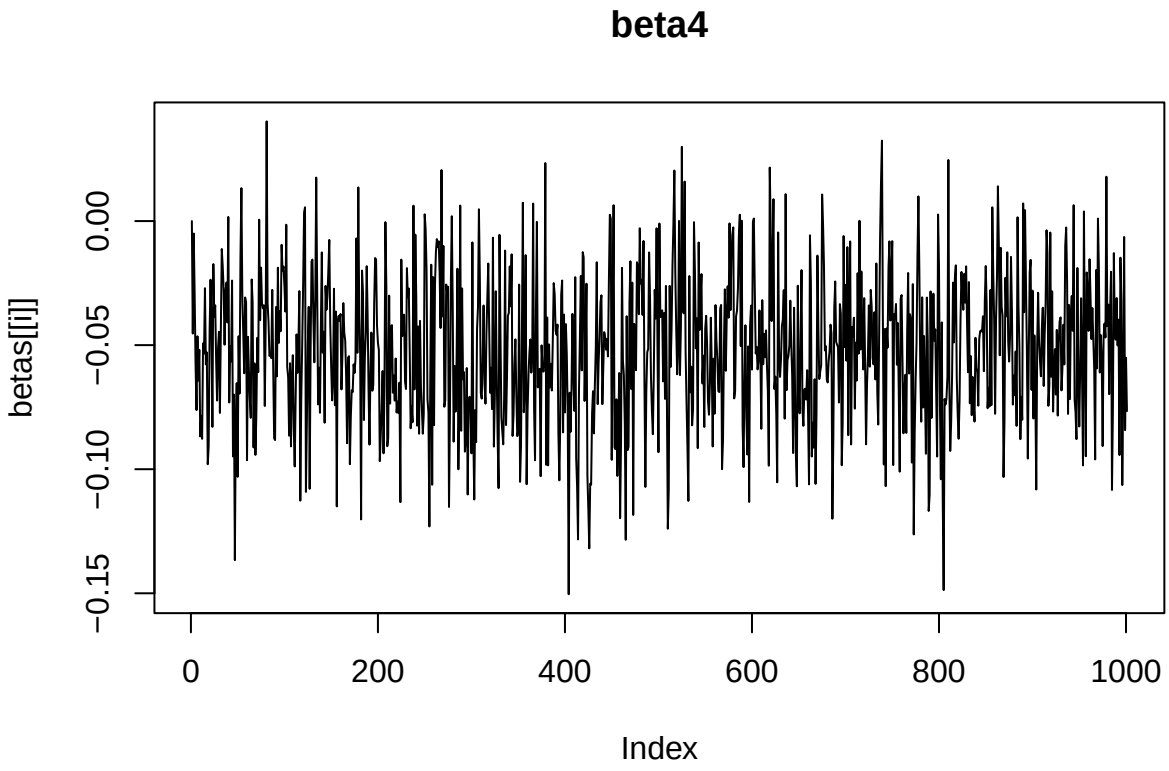
```
#Plotting the convergence:
for ( i in 1:7){
plot(betas[[i]], type = "l", main = paste0("beta",i-1))
}
```

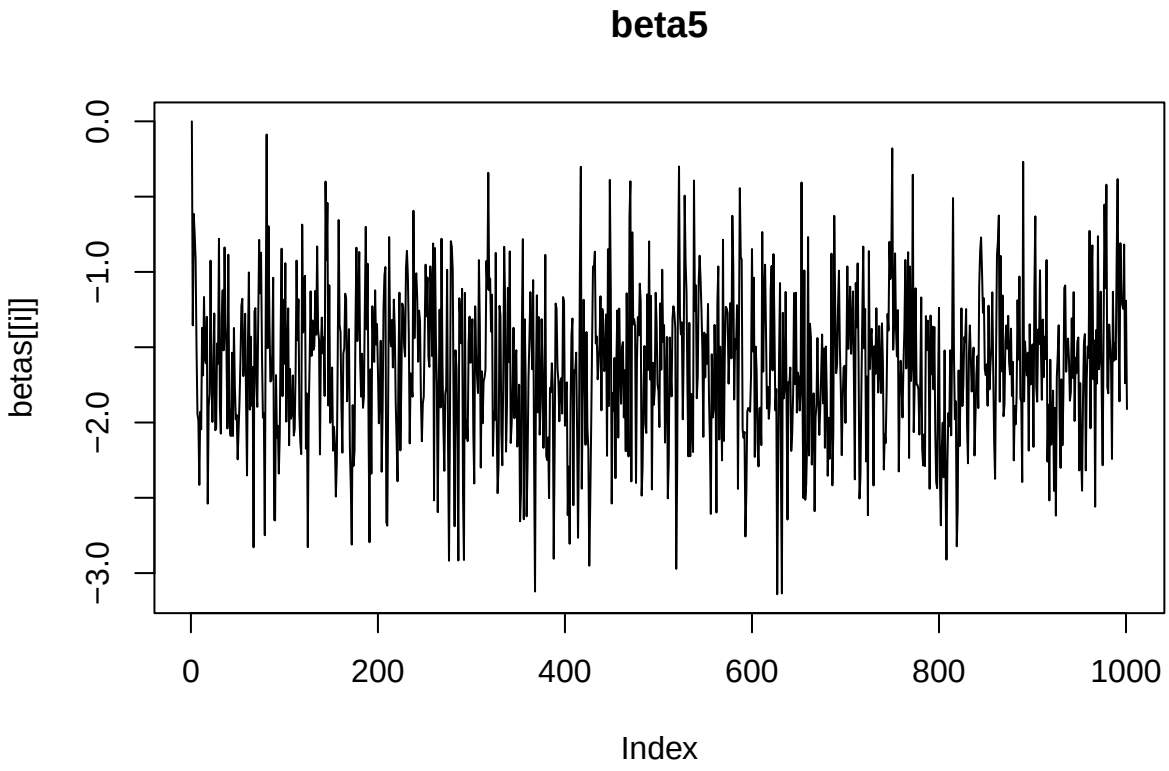

beta0
betas[[i]]
4
2
0
−2
−4
−6
0
200
400
600
800
1000
Index

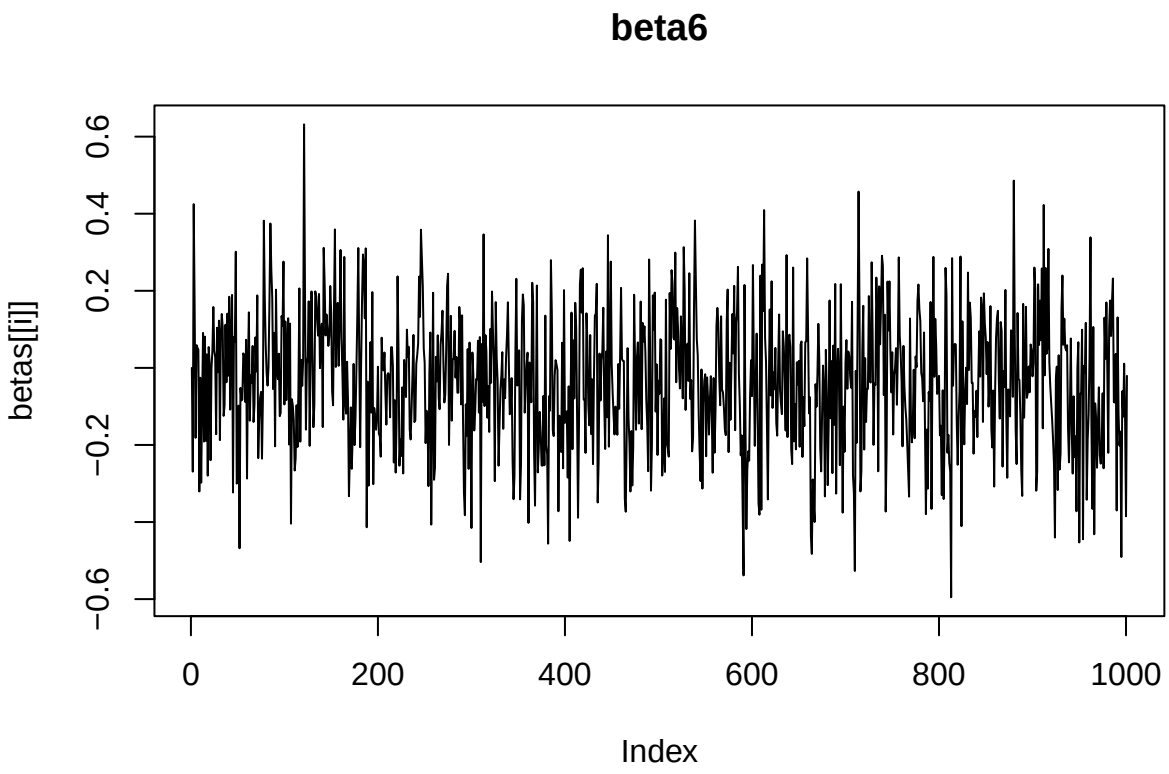

beta1
betas[[i]]
0.00
−0.05
−0.10
0
200
400
600
800
1000
Index


beta2
betas[[i]]
−0.1
0.1
0.2
0.3
0.4
0.5
0
200
400
600
800
1000
Index


beta3
betas[[i]]
0.00
0.05
0.10
0.15
0.20
0
200
400
600
800
1000
Index


beta4
betas[[i]]
0.00
−0.05
−0.10
−0.15
0
200
400
600
800
1000
Index


beta5
betas[[i]]
0.0
−1.0
−2.0
−3.0
0
200
400
600
800
1000
Index

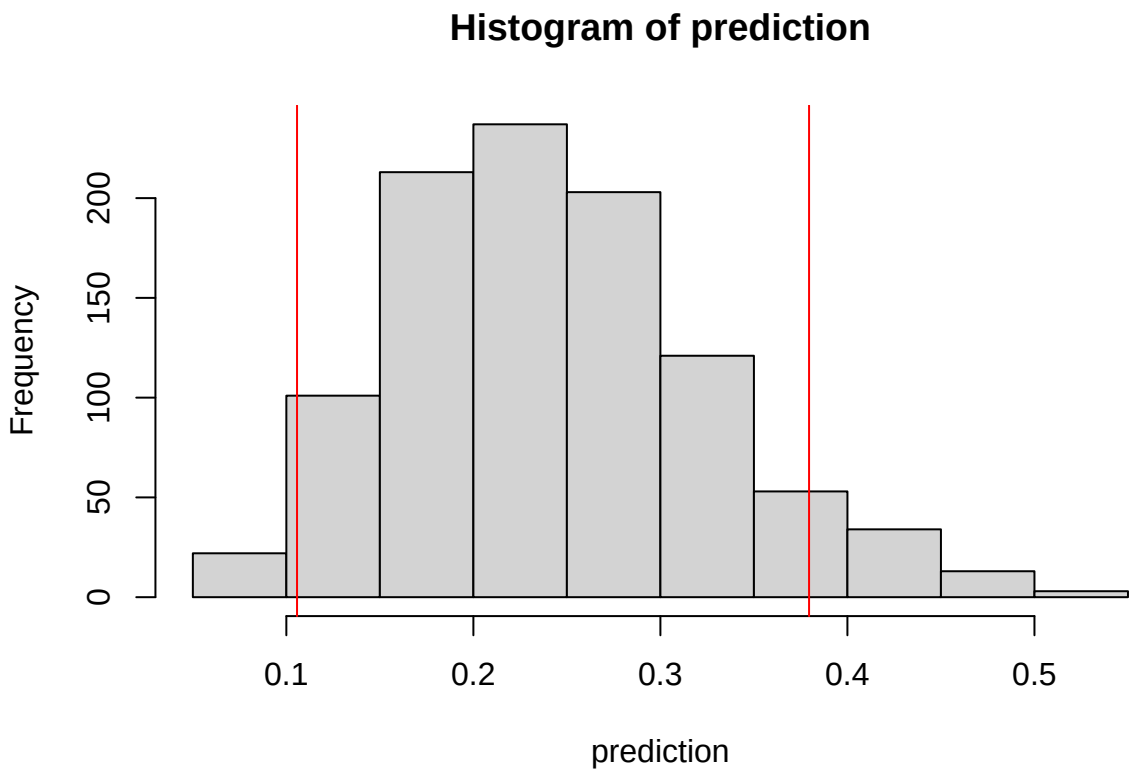


b)

Use the posterior draws from a) to compute a 90% equal tail credible interval for Pr(y = 1|x), where the values of x corresponds to a 38-year-old woman, with one child (3 years old), 12 years of education, 7 years of experience, and a husband with an income of 22. A 90% equal tail credible interval (a, b) cuts off 5% percent of the posterior probability mass to the left of a, and 5% to the right of b.

```
#Q.1) b)

x <- c(1, 22, 12, 7, 38, 1, 0)

prediction<-c()
for (i in 1:draws){
  prediction[i]<-exp(t(beta[[i]])%*%x)/(1+exp(t(beta[[i]])%*%x))
}


#plot it and find the 90% interval
hist(prediction)

low<-qnorm(0.05, mean=mean(prediction),sd=sd(prediction))
high<-qnorm(0.95, mean=mean(prediction),sd=sd(prediction))

abline(v=low, col = "red")
```

```r
abline(v=high , col = "red")
```



2 - Metropolis Random Walk for Poisson Regression

Consider the following Poisson regression model:

$$yi|\beta \sim Poisson(exp(xi^t\beta)), i = 1, ..., n,$$

where yi is the count for the ith observation in the sample and xi is the p-dimensional vector with covariate observations for the ith observation.his dataset contains observations from 800 eBay auctions of coins. The response variable is nBids and records the number of bids in each auction.

a)

Obtain the maximum likelihood estimator of beta in the Poisson regression model for the eBay data

```r
# Read data
data <- read.table("eBayNumberOfBidderData_2024.dat", header=TRUE)
X <- as.matrix(data[, 2:ncol(data)])
y <- data[, 1]

# Get maximum likelihood estimator of beta
```

```r
pois_mod <- glm(nBids ~ . - Const, family="poisson", data=data)
summary(pois_mod)
```

```
##
## Call:
## glm(formula = nBids ~ . - Const, family = "poisson", data = data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.07981    0.03393  31.828  < 2e-16 ***
## PowerSeller -0.03566    0.04167  -0.856 0.392109
## VerifyID    -0.45564    0.12748  -3.574 0.000351 ***
## Sealed       0.45515    0.06226   7.311 2.65e-13 ***
## Minblem     -0.06837    0.07198  -0.950 0.342228
## MajBlem     -0.22554    0.09525  -2.368 0.017894 *
## LargNeg      0.05382    0.06406   0.840 0.400787
## LogBook     -0.08499    0.03234  -2.628 0.008599 **
## MinBidShare -1.82490    0.07843 -23.269  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1699.6  on 799  degrees of freedom
## Residual deviance:  691.8  on 791  degrees of freedom
## AIC: 2879.1
##
## Number of Fisher Scoring iterations: 5
```

The intercept and variables VerifyID, Sealed, MajBlem, LogBook, and MinBidShare are significant.

b)

$$P(yi \mid \lambda i) = (\lambda^y * e^{-}\lambda)/y$$

Let's do a Bayesian analysis of the Poisson regression. Let the prior be:

$$\beta \sim N(0, 100 * (X^T X)^{-}1)$$

where X is the n × p covariate matrix.

Assume that the posterior density is approximately multivariate normal:

$$\beta|y \sim N(\beta, J^{-}1(\beta))$$

```r
log_post_poisson <- function(betas, X, y) {
  # Prior params
  mu <- rep(0, ncol(X))
  Sigma <- 100 * solve(t(X) %*% X)

  lin_pred <- X %*% betas
  log_lik <- sum(y * lin_pred - exp(lin_pred))
  log_prior <- dmvnorm(as.vector(betas), mu, Sigma, log=TRUE)
```

```
  return(log_lik + log_prior)
}

# Initial values for beta
init_val <- matrix(0, ncol(X))

# Optimize betas
optim_res <- optim(init_val, log_post_poisson, gr=NULL, X, y,
                   method=c("BFGS"), control=list(fnscale=-1), hessian=TRUE)
names(optim_res$par) <- colnames(X)

approx_post_std <- sqrt(diag(solve(-optim_res$hessian)))
names(approx_post_std) <- colnames(X)

# Print results
cat("Posterior mode:\n", optim_res$par,
    "\n\nApprox. posterior std.:\n", approx_post_std)
```

```
## Posterior mode:
##  1.077217 -0.03567963 -0.4535318 0.4548486 -0.06863401 -0.2258391 0.05387677 -0.08454639 -1.822757
##
## Approx. posterior std.:
##  0.03389556 0.04167562 0.127156 0.06227165 0.071983 0.09527403 0.06408047 0.03233568 0.07826924
```

c)

Let's simulate from the actual posterior of using the Metropolis algorithm and compare the results with the approximate results in b). Program a general function that uses the Metropolis algorithm to generate random draws from an arbitrary posterior density. In order to show that it is a general function for any model, we denote the vector of model parameters by theta.

Note: The first argument of your (log) posterior function should be theta, the vector of parameters for which the posterior density is evaluated

$$\theta p|\theta(i-1) \sim N(\theta(i-1), c \cdot \Sigma)$$

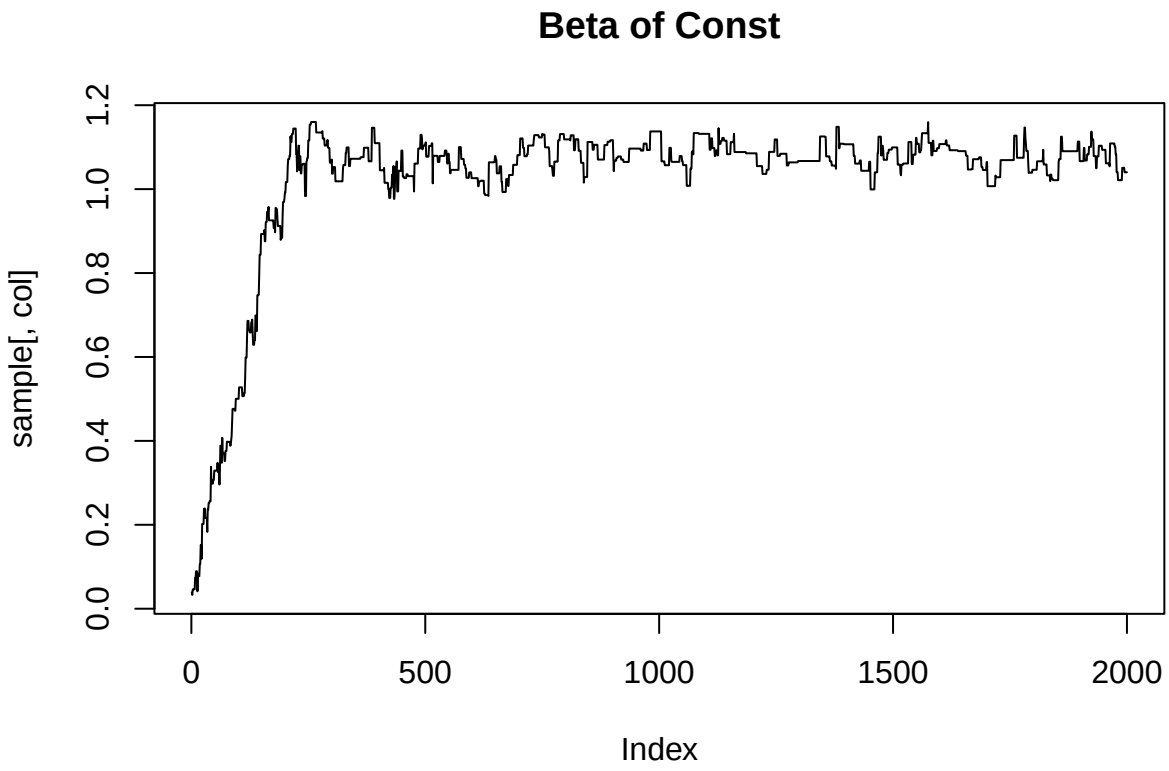
Compute the acceptance probability:

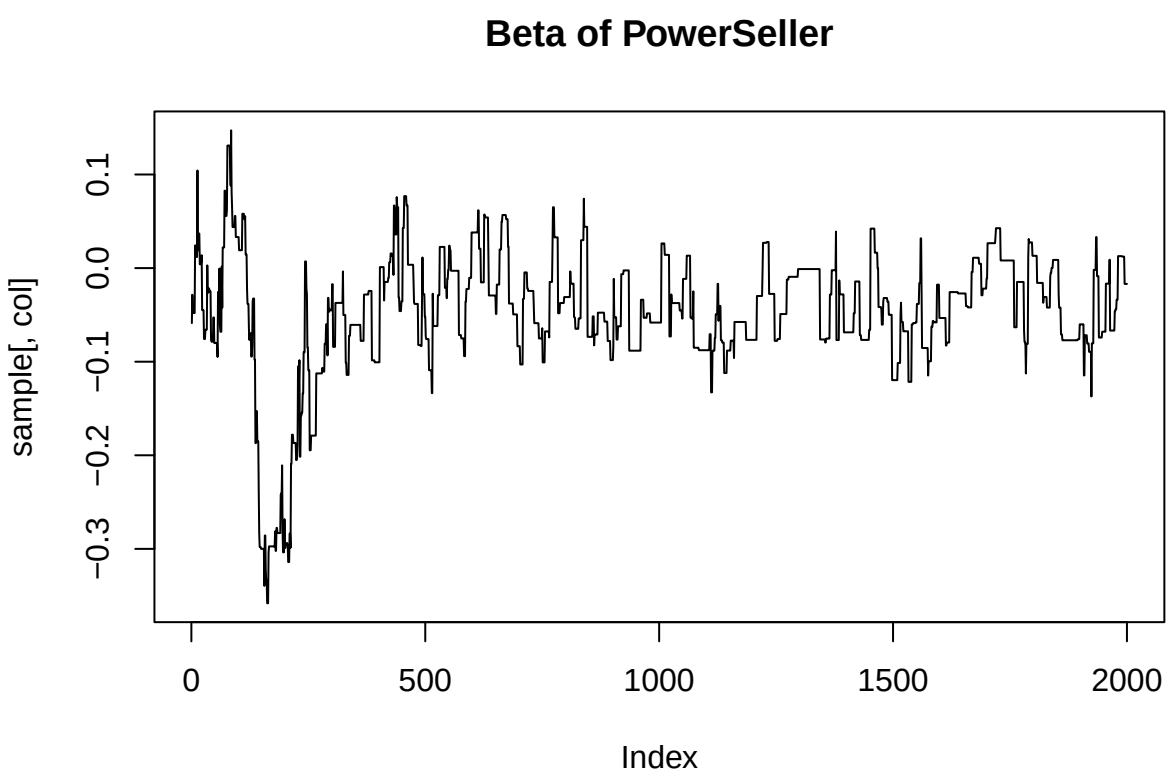
$$\alpha = min(1, p(\theta p|y)/p(\theta(i-1)|y))$$

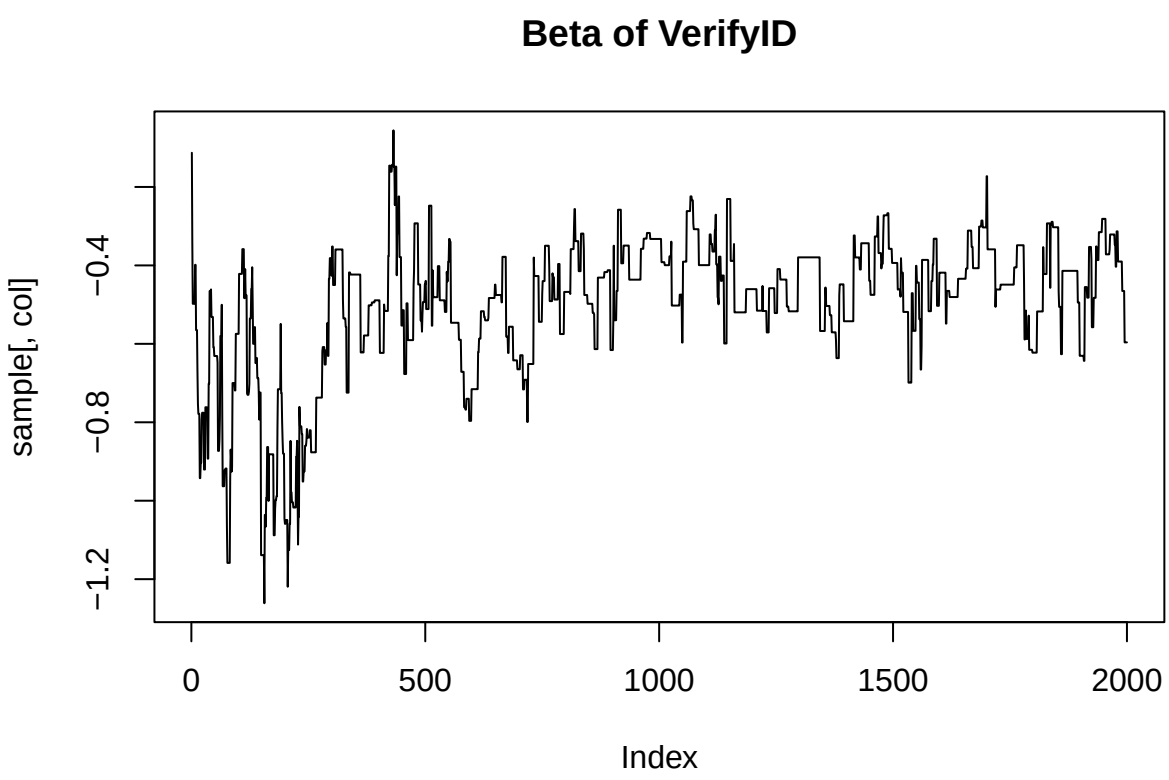
$$p(\theta p|y)/p(\theta(i-1)|y) = exp(logp(\theta p|y) - logp(\theta(i-1)|y))$$

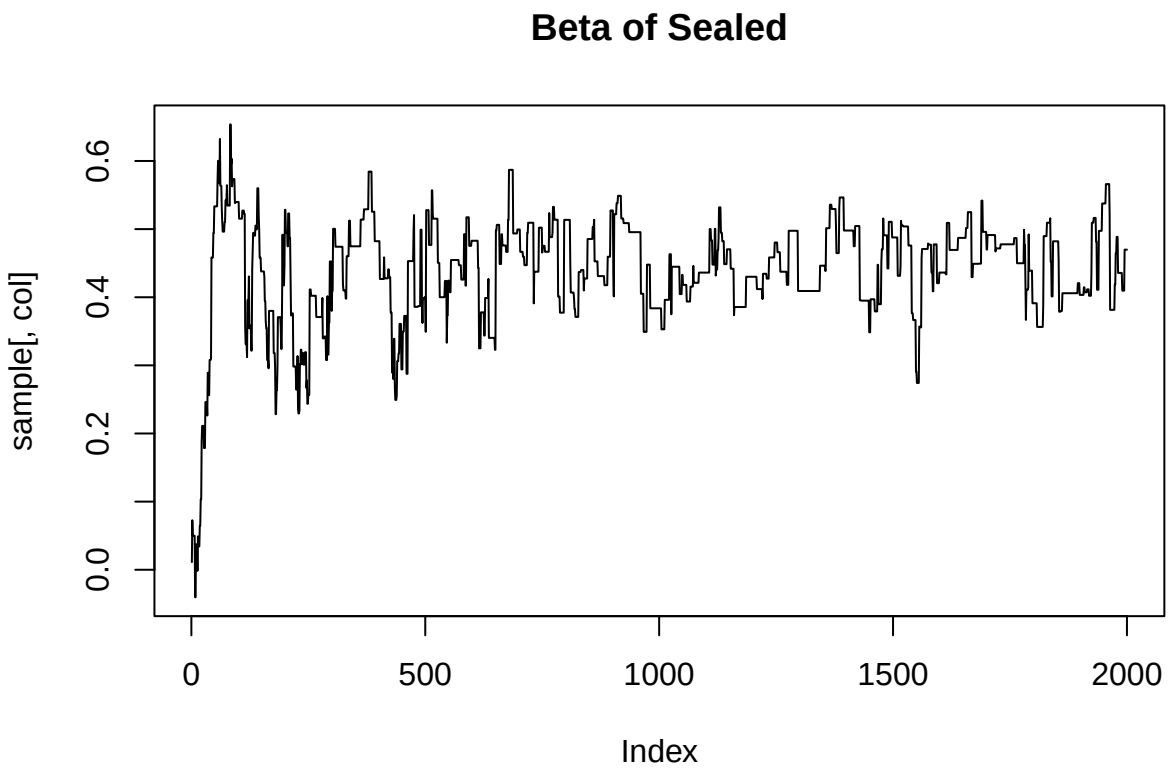
```
# Function to draw samples using MH algorithm
metropolis_hastings <- function(theta, n_samples, log_post_fun, X, y,
                                Sigma, const) {
  sample <- matrix(nrow=n_samples, ncol=nrow(theta))

  for (i in 1:n_samples) {
    sample_proposal <- t(rmvnorm(1, theta, const * Sigma))
    mh_ratio <- exp(log_post_fun(sample_proposal, X, y)
```

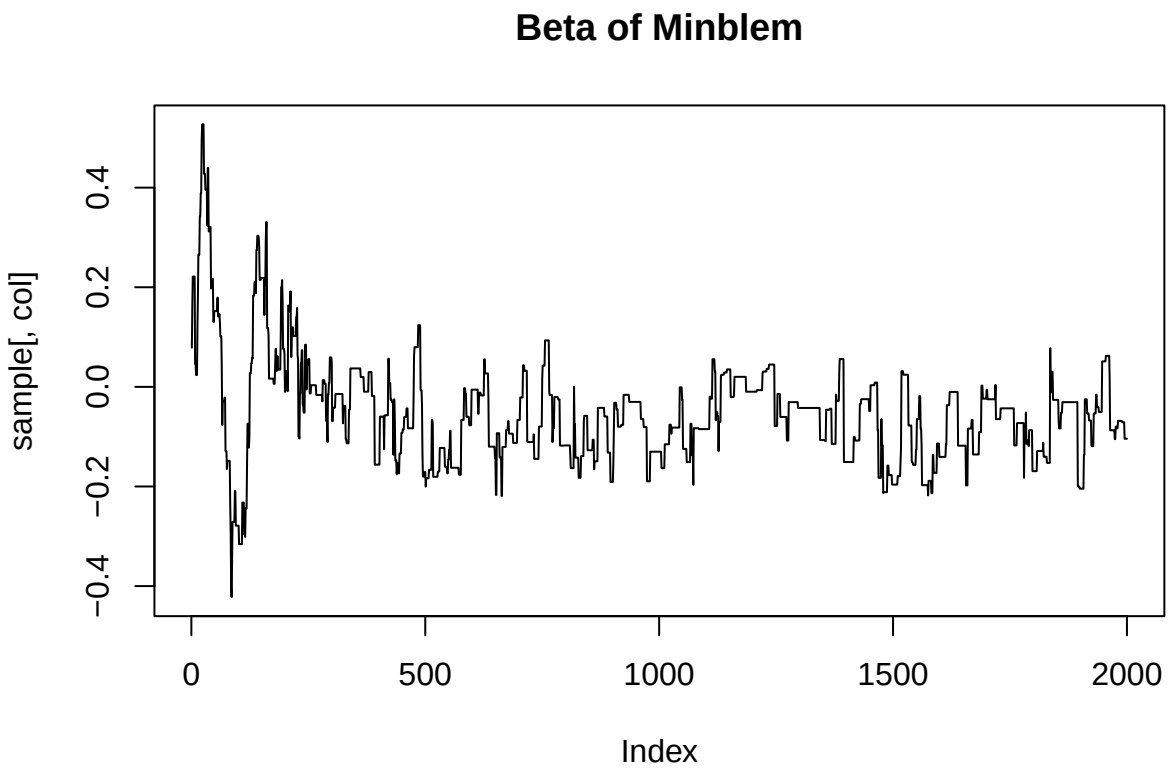
```
                    - log_post_fun(theta, X, y))
    acc_prob <- min(1, mh_ratio)

    # If accepted add proposal to sample, else add previous to sample
    if (runif(1) <= acc_prob) {
      theta <- sample_proposal
    }

    sample[i,] <- theta
  }

  return(sample)
}

# Get sample and visualize convergence
sample <- metropolis_hastings(init_val, 2000, log_post_poisson, X, y,
                              solve(-optim_res$hessian), 1)

for (col in 1:ncol(sample)) {
  plot(sample[, col], type="l", main=paste0("Beta of ", colnames(X)[col]))
}
```

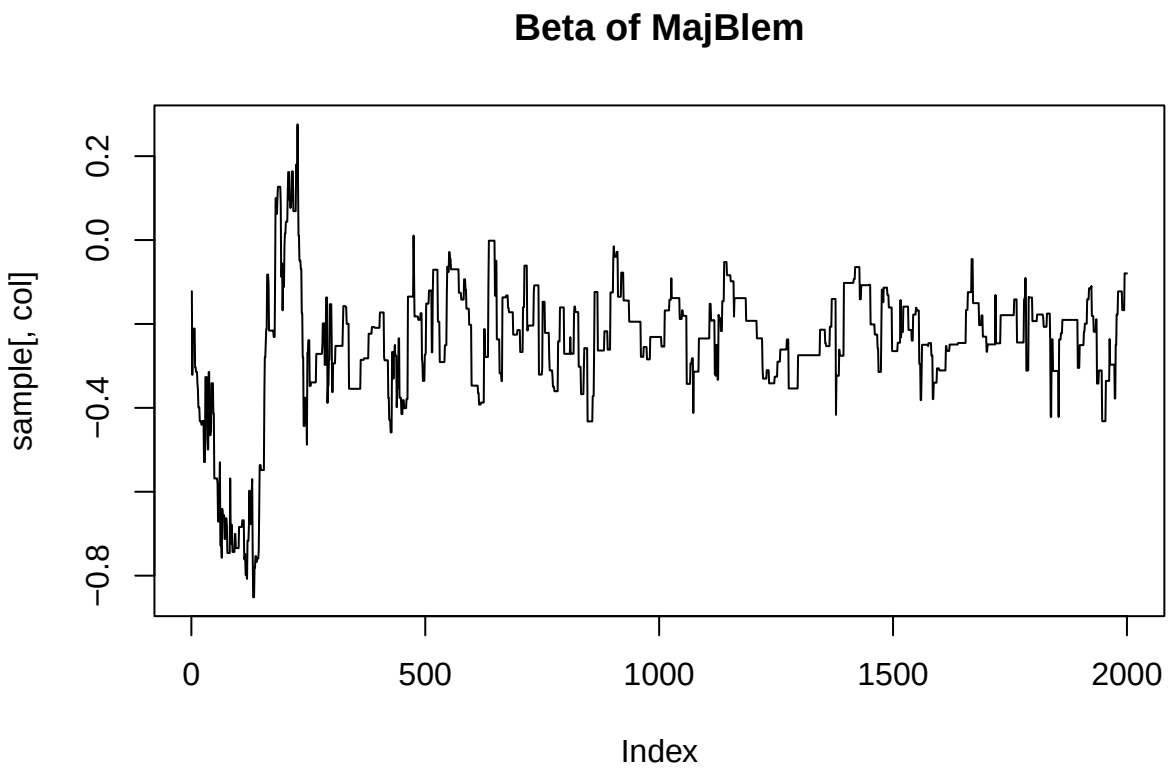


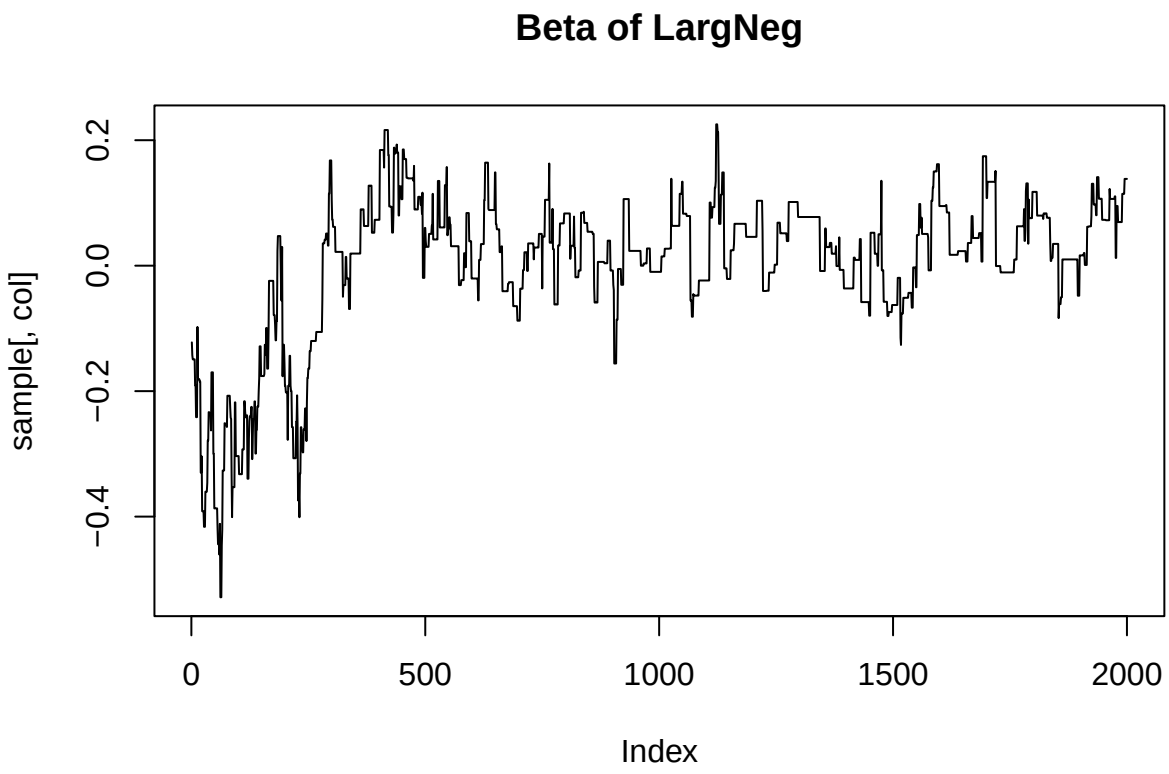

Beta of PowerSeller
sample[, col]
−0.3
−0.2
−0.1
0.0
0.1
0
500
1000
1500
2000
Index

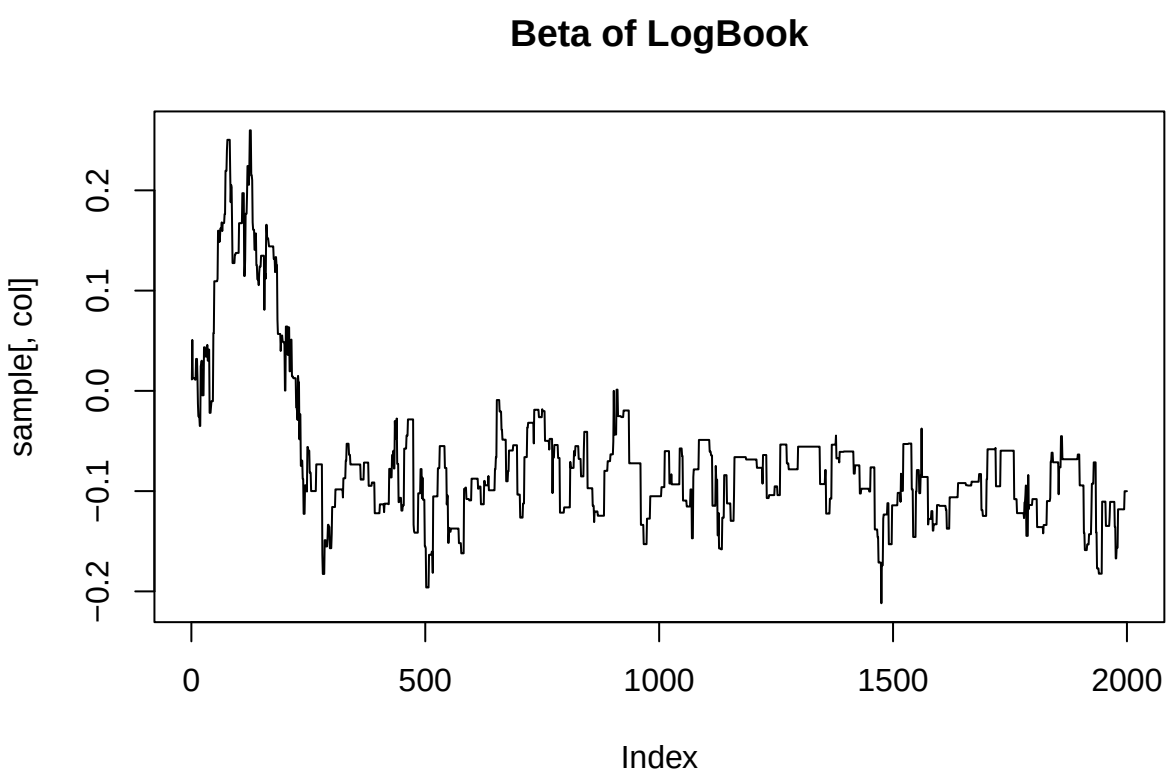

Beta of VerifyID
sample[, col]
−1.2
−0.8
−0.4
0
500
1000
1500
2000
Index

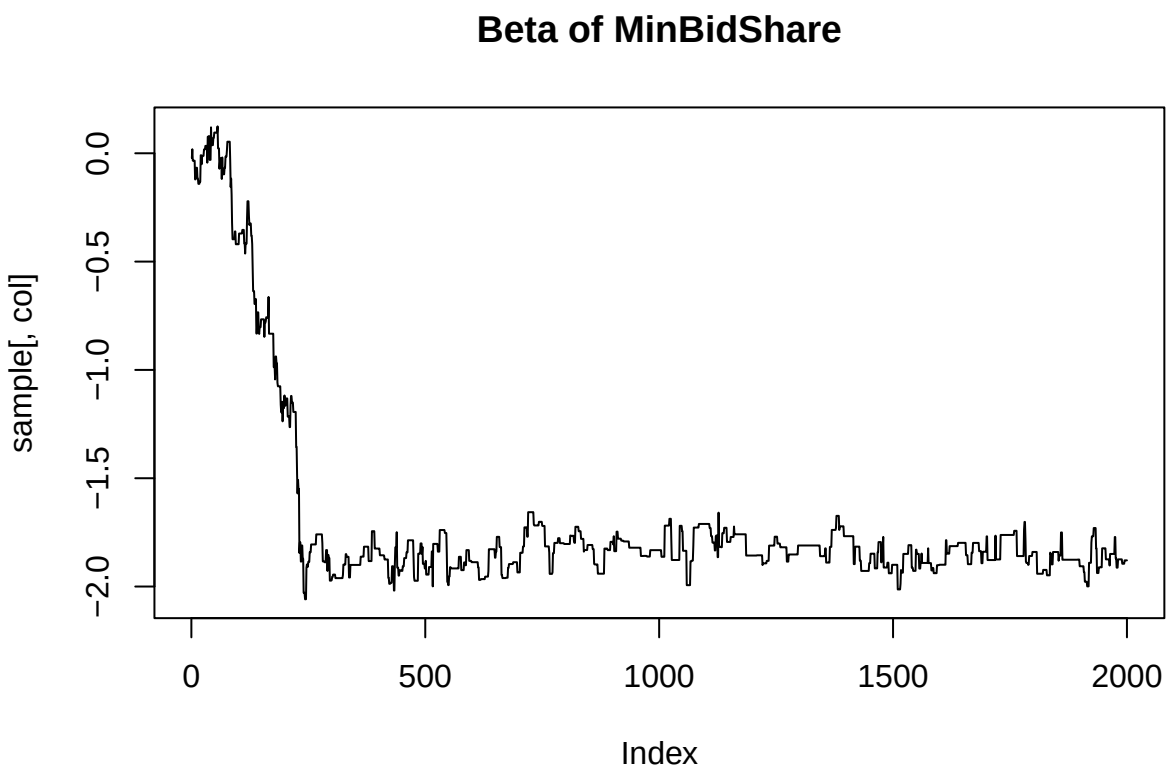

Beta of Sealed
sample[, col]
0.0
0.2
0.4
0.6
0
500
1000
1500
2000
Index


Beta of Minblem
sample[, col]
−0.4
−0.2
0.0
0.2
0.4
0
500
1000
1500
2000
Index


Beta of MajBlem
sample[, col]
0.2
0.0
−0.4
−0.8
0
500
1000
1500
2000
Index


Beta of LargNeg
sample[, col]
0.2
0.0
−0.2
−0.4
0
500
1000
1500
2000
Index

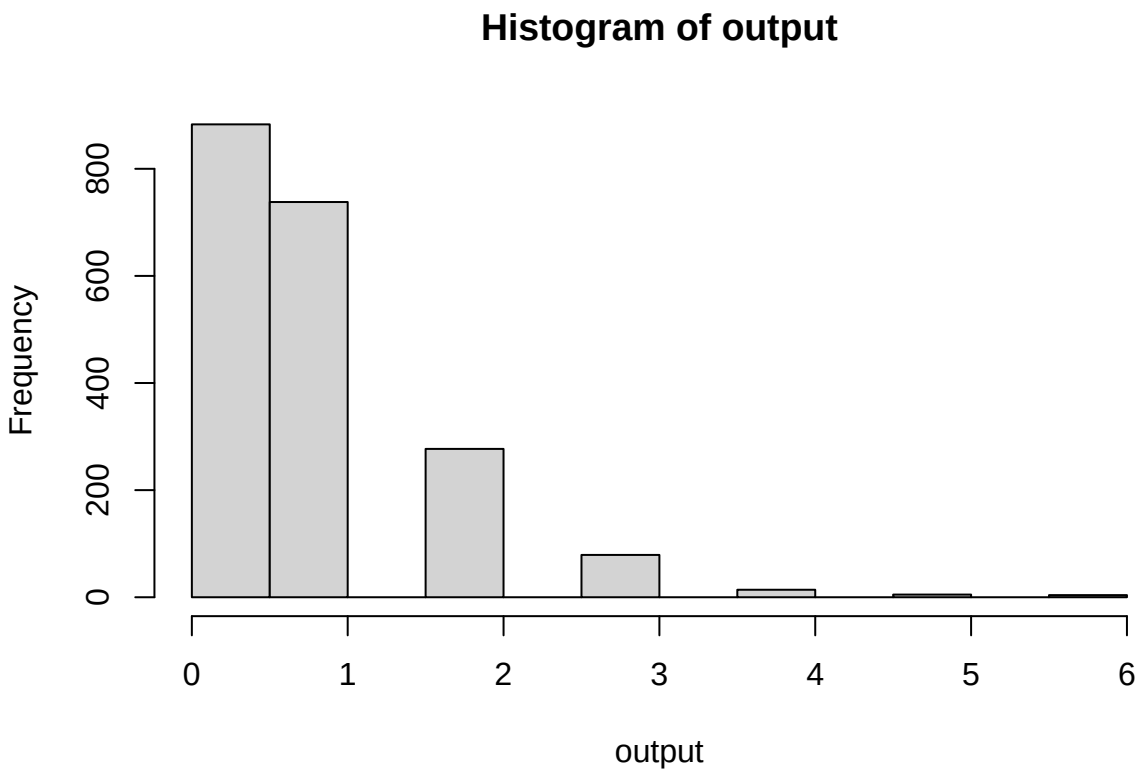

Beta of LogBook
sample[, col]
−0.2
−0.1
0.0
0.1
0.2
0
500
1000
1500
2000
Index



d)

Use the MCMC draws from c) to simulate from the predictive distribution of the number of bidders in a new auction with the characteristics below. Plot the predictive distribution. What is the probability of no bidders in this new auction?

```
x <- c(1, 1, 0, 1, 0, 1, 0, 1.2, 0.8)
y_pred <- exp(sample %*% x)


output<-c()
for (i in 1:length(y_pred)){
  output[i]<-rpois(1,y_pred[i])
}
hist(output)
```




```
# Prob. of no bidders
p_no_bid <- sum(output < 0.5) / length(output)
cat("Pr(no bidders) =", round(p_no_bid, 3))
```

```
## Pr(no bidders) = 0.442
```

Assuming that a predicted count of less than 0.5 means the number of bidders will be 0.

#Question-3

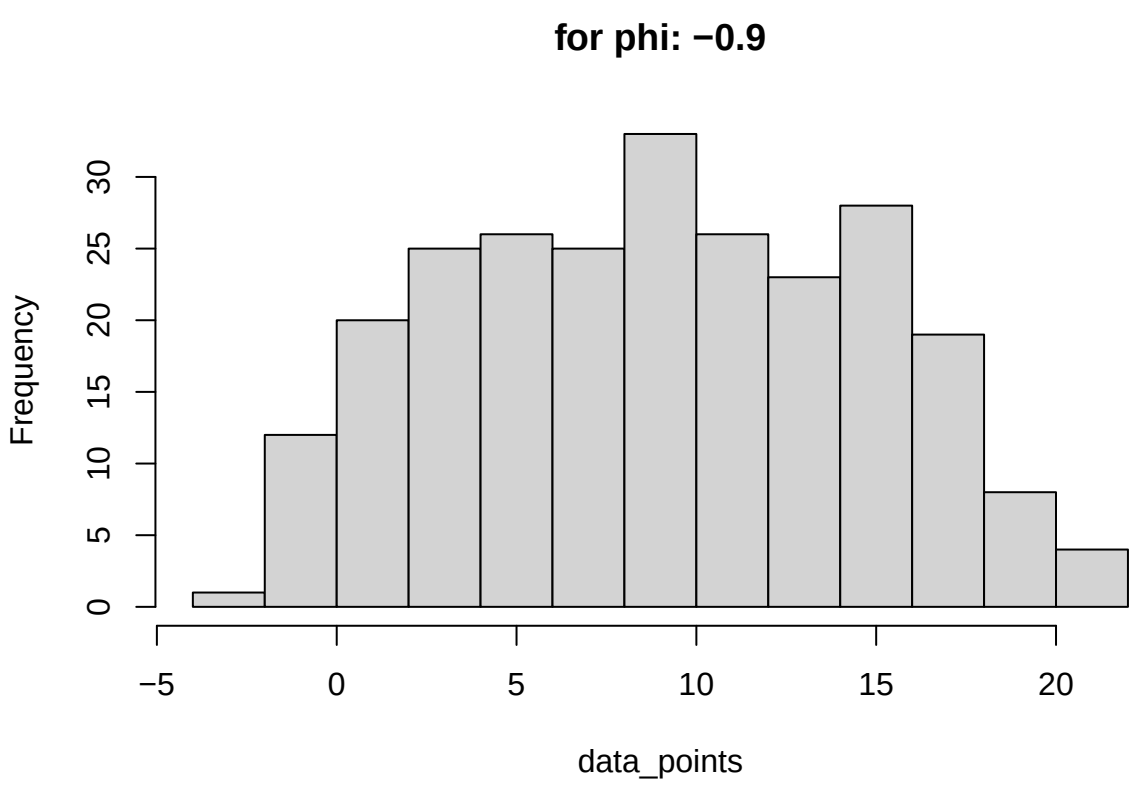
a)

Write a function in R that simulates data from the AR(1)-process:

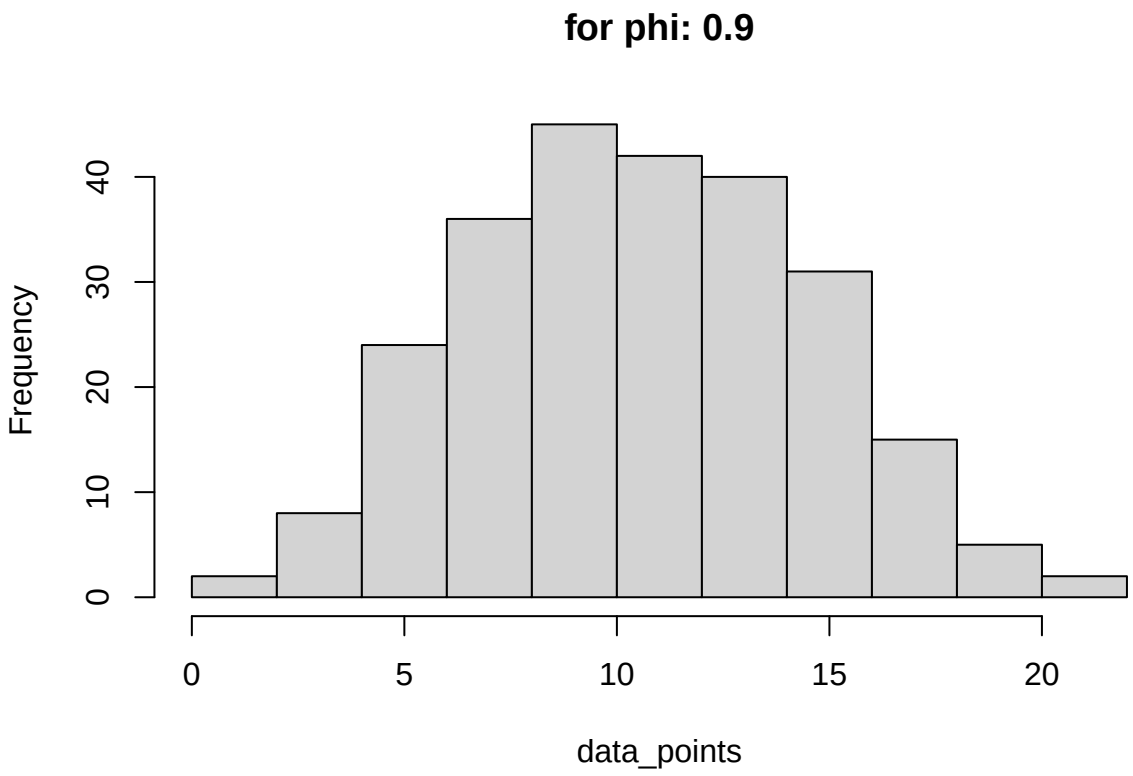
$$xt = \mu + \phi * (xt - 1 - \mu) + \varepsilon t, \varepsilon t \sim N(0, \sigma^2)$$

Start the process at x1 = mu and then simulate values for xt for t = 2, 3 . . . , T and return the vector x1:T containing all time points. Use mu = 9, sigma^2 = 4 and T = 250 and look at some different realizations (simulations) of x1:T for values of phi between −1 and 1 (this is the interval of phi where the AR(1)-process is stationary). Include a plot of at least one realization in the report. What effect does the value of phi have on x1:T ?

```r
#Q.3) a)

mu <- 9
sigma <- 2
T <- 250

phi<- runif(1,min = -1,max = 1)

data_points<-c()
data_points[1]<- mu

AR <- function(mu,sigma_squared,phi,T){
  for (i in 2:T){
   data_points[i]<<-mu + phi*(data_points[i-1] - mu) + rnorm(1,mean = 0, sd = 2)
  }
}

AR(mu,sigma_squared,-0.9,T)
hist(data_points, main = paste("for phi:",-0.9))
```




```r
AR(mu,sigma_squared,0.9,T)
hist(data_points, main = paste("for phi:",0.9))
```



As we move the phi values from -1 to 1, the data becomes more concentrated to particular values as we can see from the above histograms.

#b) Use your function from a) to simulate two AR(1)-processes, x1:T with phi = 0.3 and y1:T with phi = 0.97. Now, treat your simulated vectors as synthetic data, and treat the values of mu, phi and sigma^2 as unknown parameters. Implement Stan code that samples from the posterior of the three parameters, using suitable non-informative priors of your choice.

The <lower=0> constraint ensures that N cannot be negative.

#If dont explictly write " mu ~ uniform(-10, 10);" , it will assume that the parameter is uniform.

#i) Report the posterior mean, 95% credible intervals and the number of effective posterior samples for the three inferred parameters for each of the simulated AR(1)-process. Are you able to estimate the true values?

```
#i)


#for the 1st data points:
cat("for phi = 0.3 data points\n")
```

```
## for phi = 0.3 data points
```

```
cat("mu:","(posterior mean:",values1[1,1],")\n")
```

```
## mu: (posterior mean: 9.251381 )
```

```
cat("mu:","(95% interval:",paste0("[", round(qnorm(0.025, mean=mean(postDraws$mu),sd=sd(postDraws$mu)),
```

```
## mu: (95% interval: [8.858, 9.645] )
```

```
cat("mu:","(effective samples:",values1[1,9],")\n")
```

```
## mu: (effective samples: 4412.549 )
```

```
cat("phi:","(posterior mean:",values1[2,1],")\n")
```

```
## phi: (posterior mean: 0.3443144 )
```

```
cat("phi:","(95% interval:",paste0("[", round(qnorm(0.025, mean=mean(postDraws$phi),sd=sd(postDraws$phi
```

```
## phi: (95% interval: [0.225, 0.463] )
```

```
cat("phi:","(effective samples:",values1[2,9],")\n")
```

```
## phi: (effective samples: 3705.507 )
```

```
cat("sigma2:","(posterior mean:",values1[3,1],")\n")
```

```
## sigma2: (posterior mean: 4.449419 )
```

```
cat("sigma2:","(95% interval:",paste0("[", round(qnorm(0.025, mean=mean(postDraws$sigma2),sd=sd(postDrau
```

```
## sigma2: (95% interval: [3.661, 5.238] )
```

```
cat("sigma2:","(effective samples:",values1[3,9],")\n")
```

```
## sigma2: (effective samples: 4246.158 )
```

```
cat("epsilon:","(posterior mean:",values1[4,1],")\n")
```

```
## epsilon: (posterior mean: -2.788951e+17 )
```

```
cat("epsilon:","(95% interval:",paste0("[", round(qnorm(0.025, mean=mean(postDraws$epsilon),sd=sd(postD
```

```
## epsilon: (95% interval: [-7358035034478278656, 6800244735818653696] )
```

```
cat("epsilon:","(effective samples:",values1[4,9],")\n")
```

```
## epsilon: (effective samples: 3.448646 )
```

```r
cat("\n")
```

```r
cat("\n")
```

```r
#For the 2nd data points
cat("for phi = 0.97 data points\n")
```

```
## for phi = 0.97 data points
```

```r
cat("mu:","(posterior mean:",values2[1,1],")\n")
```

```
## mu: (posterior mean: 3.362395 )
```

```r
cat("mu:","(95% interval:",paste0("[", round(qnorm(0.025, mean=mean(postDraws1$mu),sd=sd(postDraws1$mu)
```

```
## mu: (95% interval: [-6.663, 13.388] )
```

```r
cat("mu:","(effective samples:",values2[1,9],")\n")
```

```
## mu: (effective samples: 288.1949 )
```

```r
cat("phi:","(posterior mean:",values2[2,1],")\n")
```

```
## phi: (posterior mean: 0.9834896 )
```

```r
cat("phi:","(95% interval:",paste0("[", round(qnorm(0.025, mean=mean(postDraws1$phi),sd=sd(postDraws1$p
```

```
## phi: (95% interval: [0.962, 1.005] )
```

```r
cat("phi:","(effective samples:",values2[2,9],")\n")
```

```
## phi: (effective samples: 374.7906 )
```

```r
cat("sigma2:","(posterior mean:",values2[3,1],")\n")
```

```
## sigma2: (posterior mean: 4.126453 )
```

```r
cat("sigma2:","(95% interval:",paste0("[", round(qnorm(0.025, mean=mean(postDraws1$sigma2),sd=sd(postDra
```

```
## sigma2: (95% interval: [3.393, 4.86] )
```

```r
cat("sigma2:","(effective samples:",values2[3,9],")\n")
```

```
## sigma2: (effective samples: 743.0955 )
```

```
cat("epsilon:","(posterior mean:",values2[4,1],")\n")
```

```
## epsilon: (posterior mean: 159431.6 )
```

```
cat("epsilon:","(95% interval:",paste0("[", round(qnorm(0.025, mean=mean(postDraws1$epsilon),sd=sd(postl
```

```
## epsilon: (95% interval: [-378860.401, 697723.64] )
```

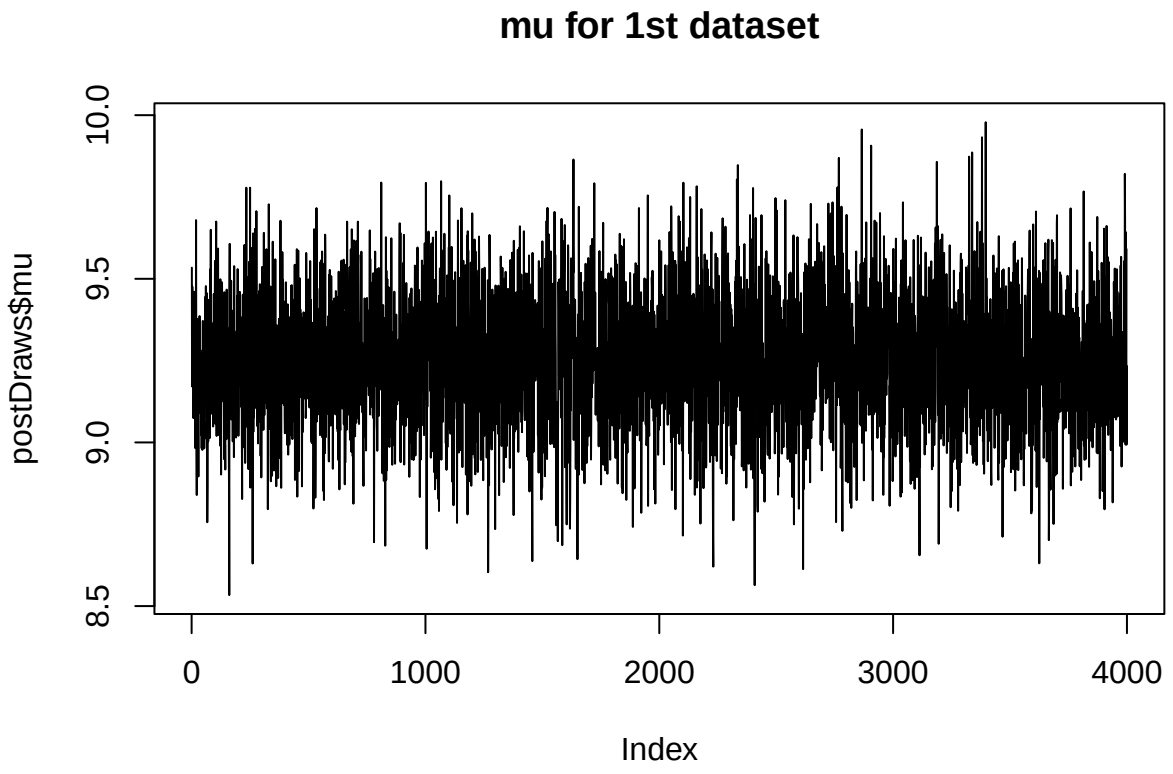
```
cat("epsilon:","(effective samples:",values2[4,9],")\n")
```

```
## epsilon: (effective samples: 3.03412 )
```

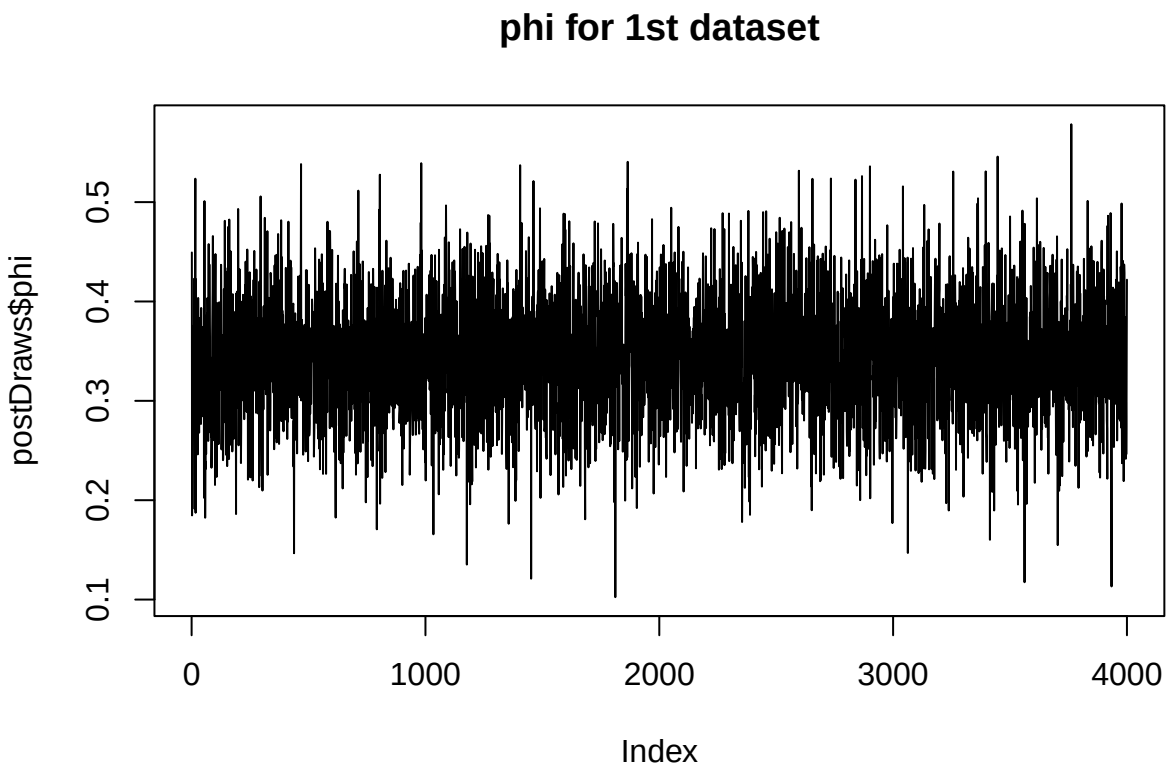
As we can see above, we are able get similar values for phi and mu to the true values for both the datasets while the sigma2 values for both the datasets can be seen to differ from the true value, this could be cause the data might have some noise.

#ii) For each of the two data sets, evaluate the convergence of the samplers and plot the joint posterior of mu and phi. Comments?

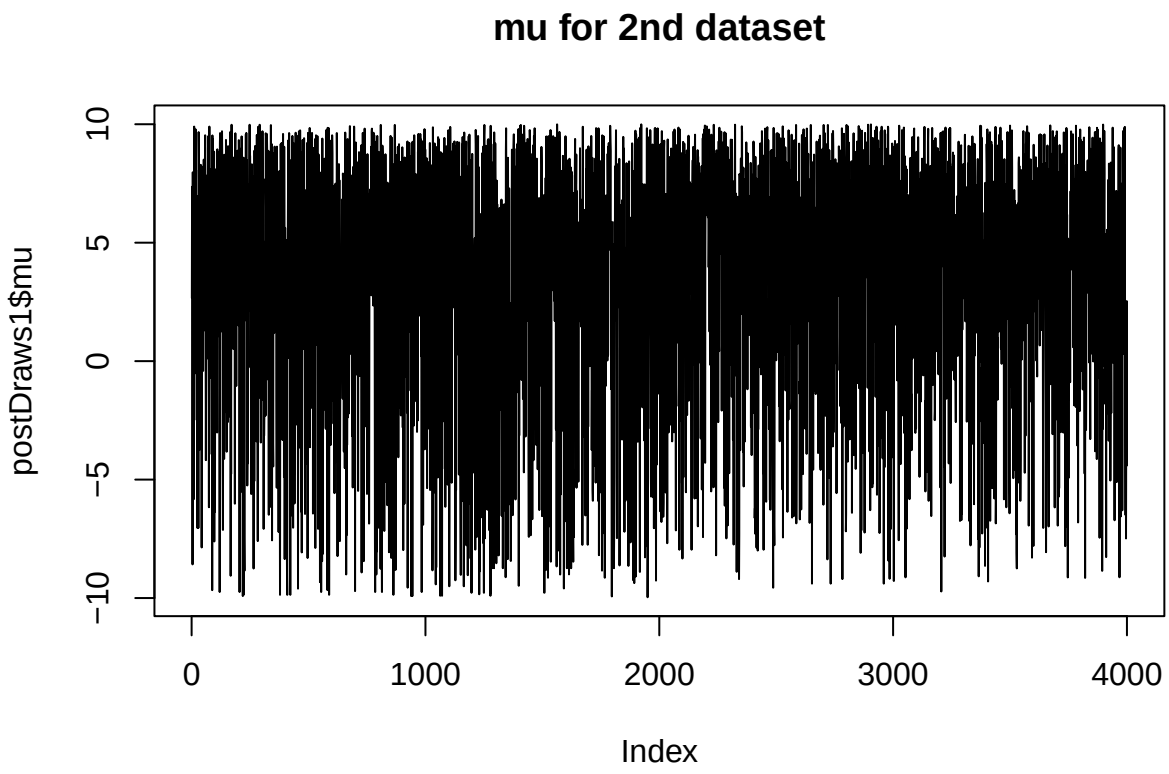
```
#For the 1st data points
plot(postDraws$mu, type = "l",main = "mu for 1st dataset")
```



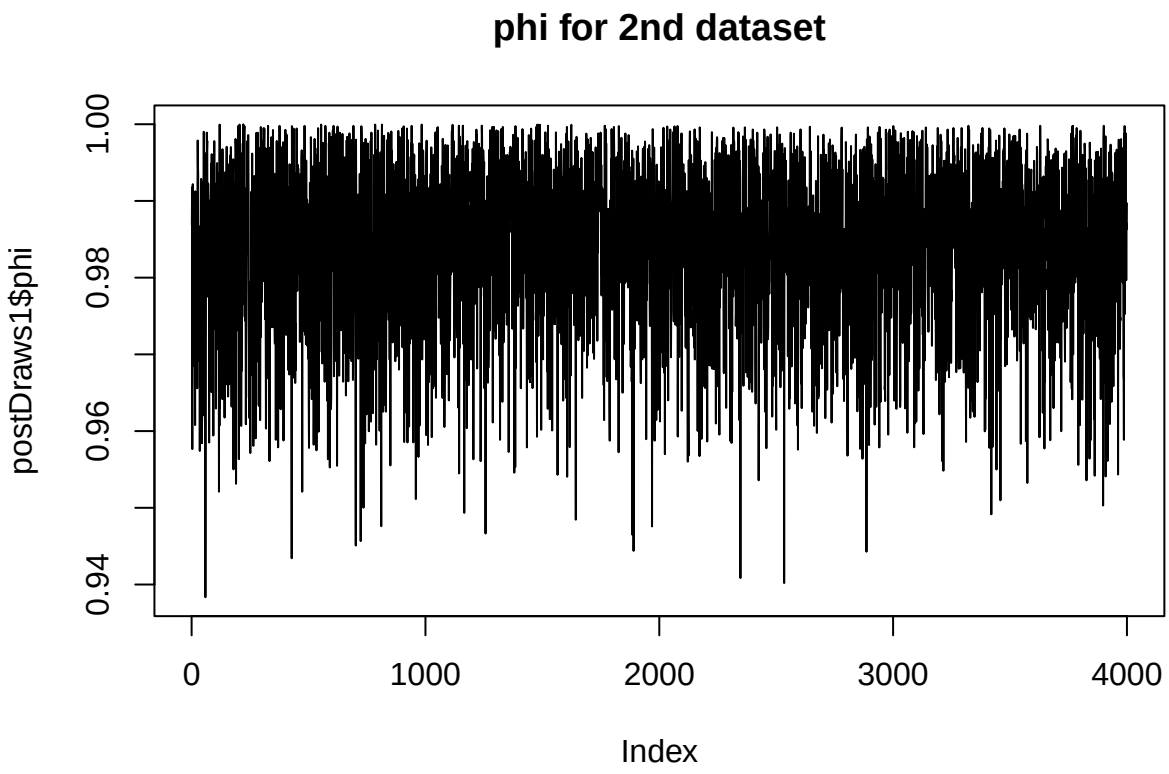
```r
plot(postDraws$phi, type = "l",main = "phi for 1st dataset")
```



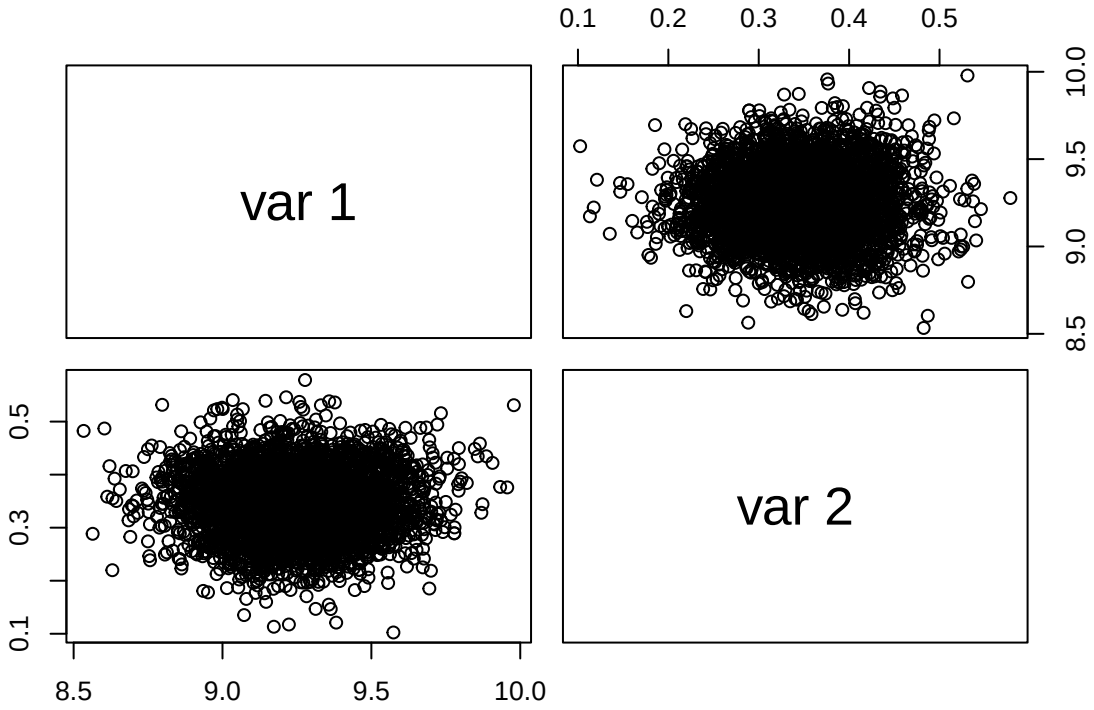

```r
#For the 2nd data points
plot(postDraws1$mu, type = "l",main = "mu for 2nd dataset")
```



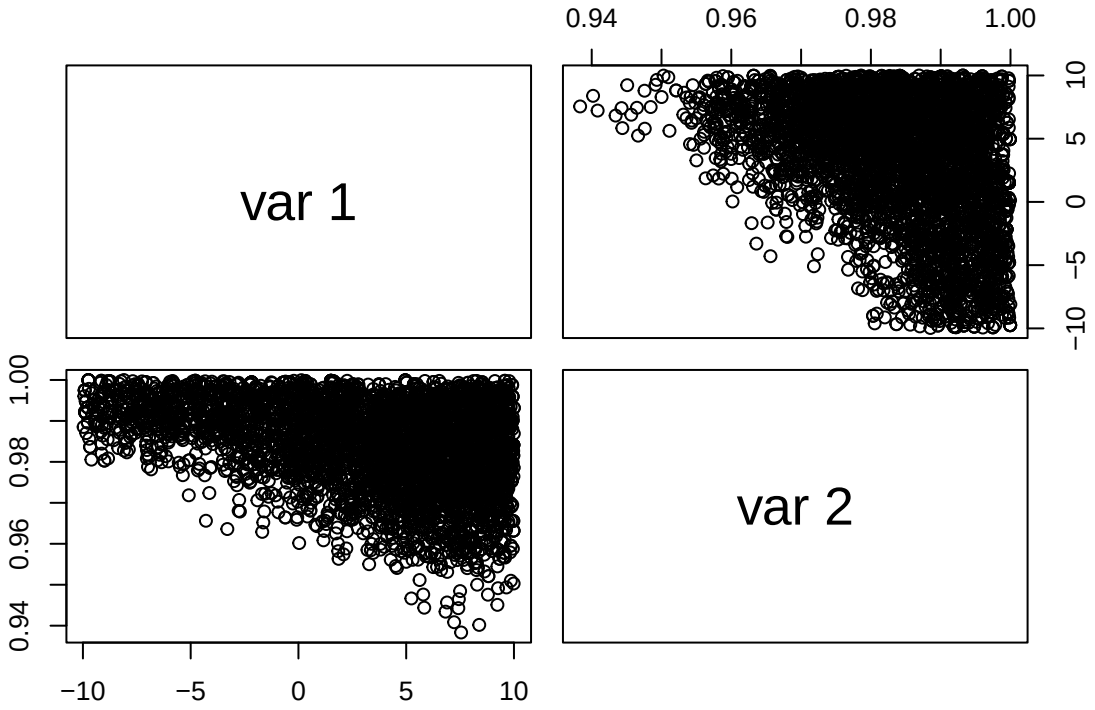

```
plot(postDraws1$phi, type = "l",main = "phi for 2nd dataset")
```




```
pairs(cbind(postDraws$mu,postDraws$phi))
```




```
pairs(cbind(postDraws1$mu,postDraws1$phi))
```



We can see that there is convergence for the 2 parameters. For the first dataset, for the range of phi values, the mu value is hovering around 9.

For the second dataset, for the range of phi values, the mu value is hovering around 9 as well.