Computer Lab 3
732A54 - Bayesian Learning

Jooyoung Lee - `joole336`
Zuxiang Li - `zuxli371`

May 14, 2020

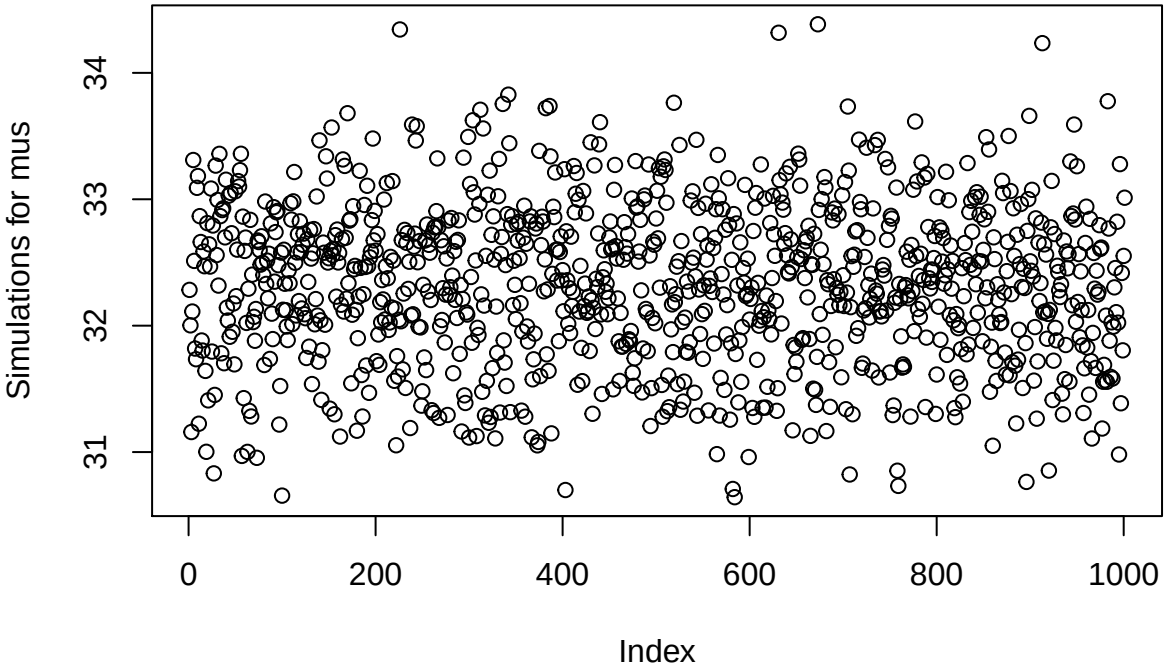
1. Normal model, mixture of normal model with semi-conjugate prior.

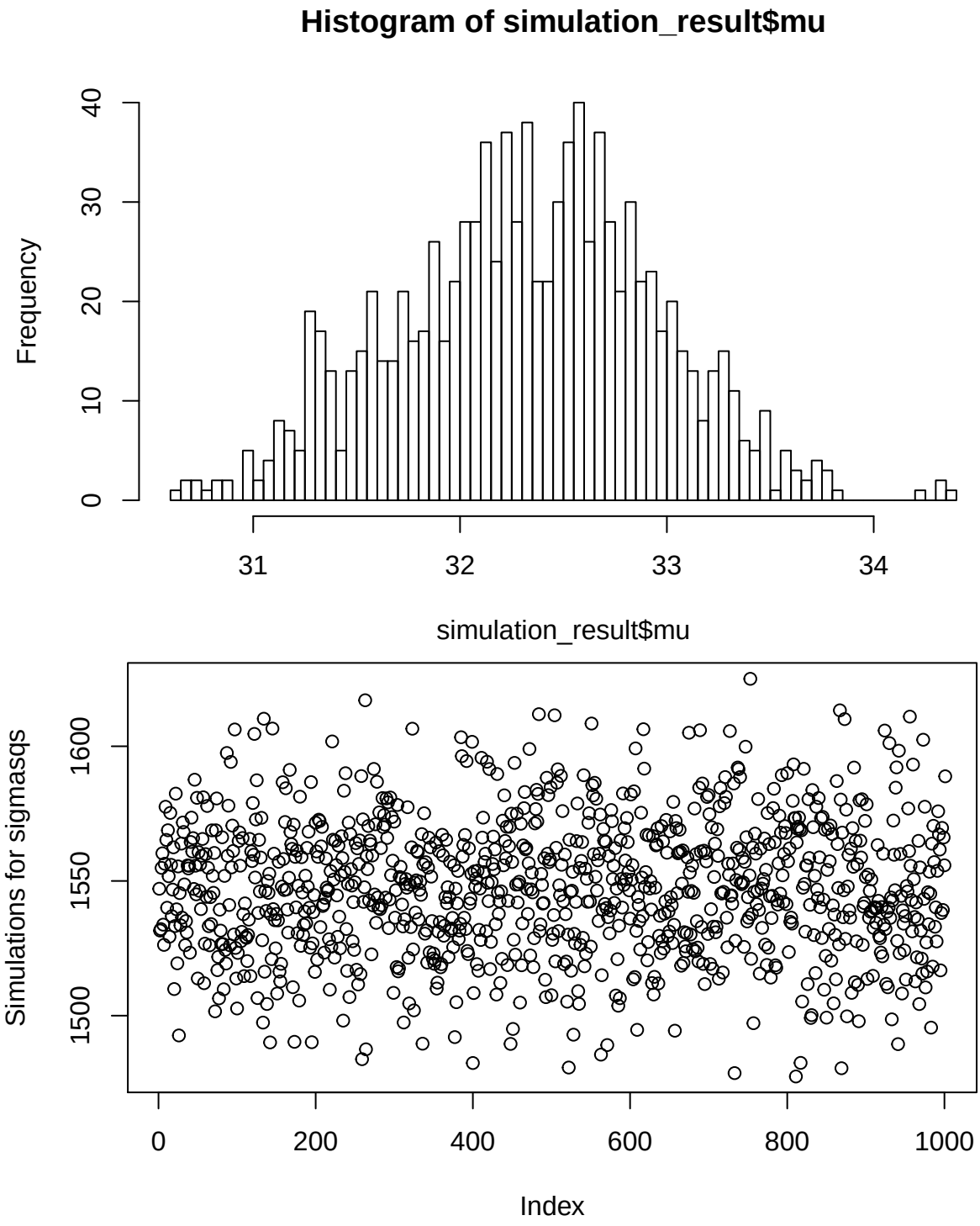
The data rainfall.dat consist of daily records, from the beginning of 1948 to the end of 1983, of precipitation (rain or snow in units of $\frac{1}{100}$ inch, and records of zero precipitation are excluded) at Snoqualmie Falls, Washington. Analyze the data using the following two models.

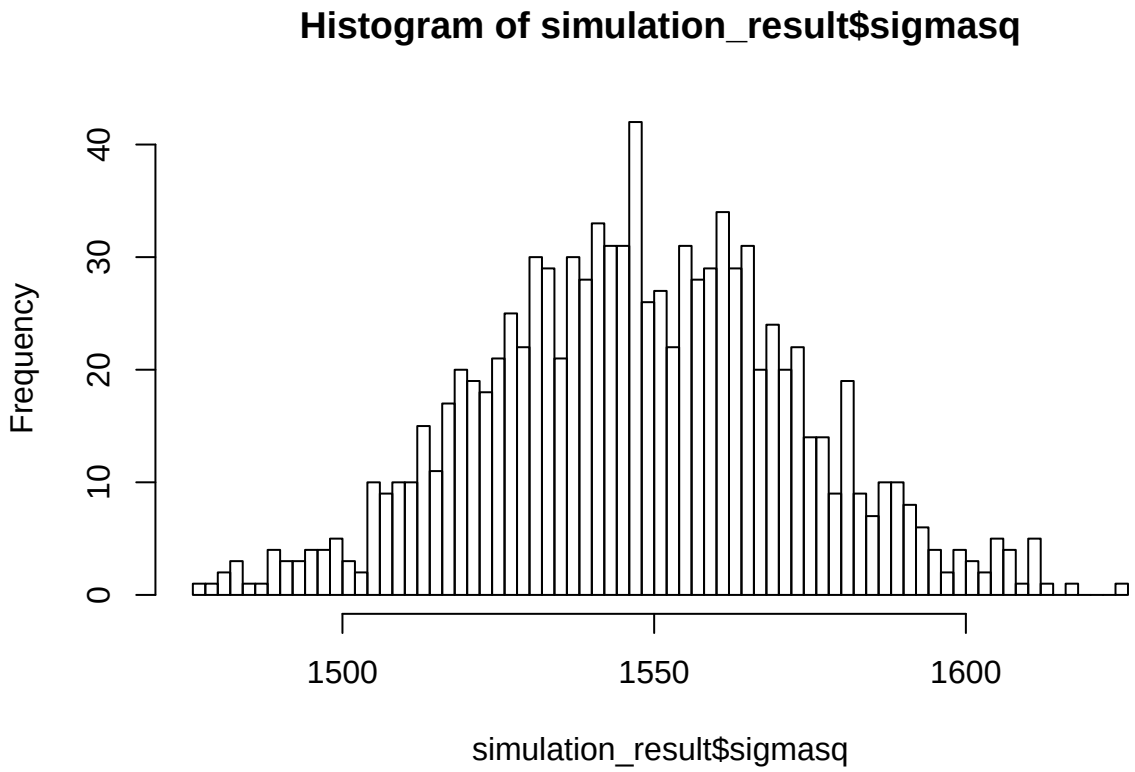
(a) Normal model. Assume the daily precipitation$\{y_1, \cdots, y_n\}$ are independent normally distributed, $y_1, \cdots, y_n|\mu, \sigma^2$ $N(\mu, \sigma^2)$ where both μ and σ^2 are unknown. Let μ $N(\mu_0, \tau^2)$ independently of $\sigma^2 \sim Inv - \chi^2(v_0, \sigma^2)$

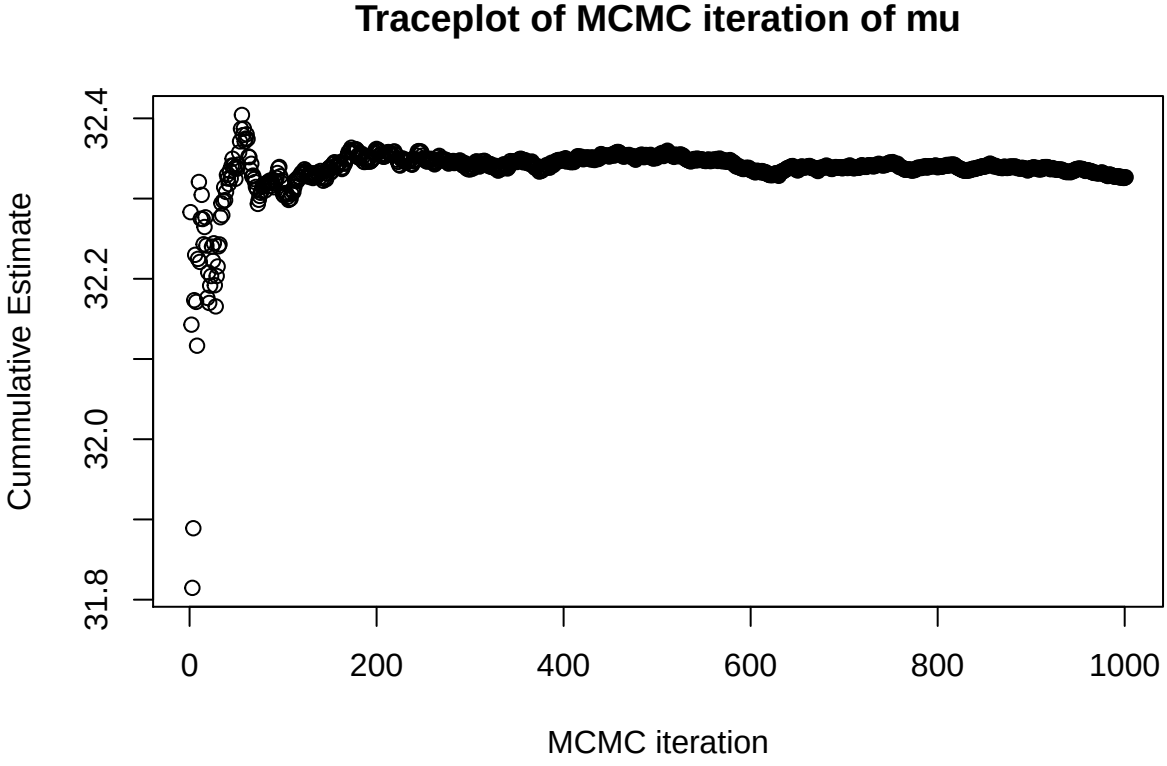
i. Implement (code!) a Gibbs sampler that simulates from the joint posterior $p(\mu, \sigma^2|y_1, \ldots, y_n)$ The full conditional posteriors are given on the slides from Lecture 7.

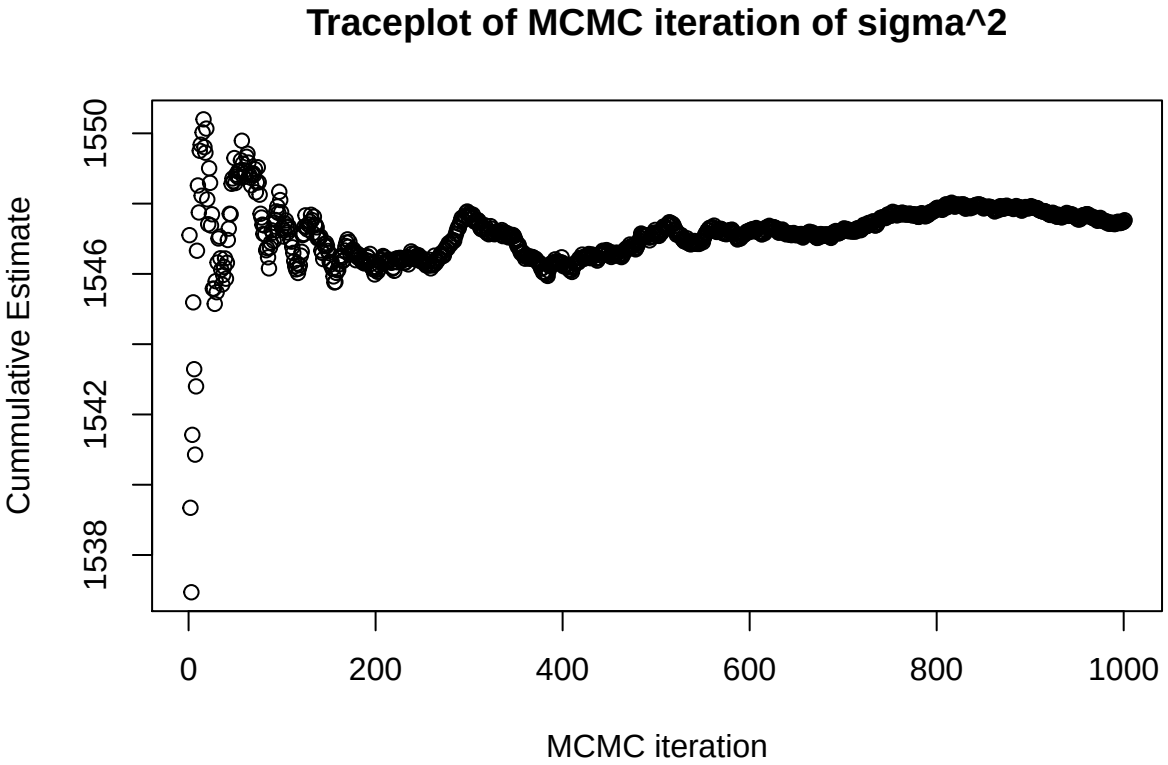
ii. Analyze the daily precipitation using your Gibbs sampler in (a)-i. Evaluate the convergence of the Gibbs sampler by suitable graphical methods, for example by plotting the trajectories of the sampled Markov chains.




Histogram of simulation_result$mu
Frequency
0
10
20
30
40
31
32
33
34
simulation_result$mu
Simulations for sigmasqs
1500
1550
1600
0
200
400
600
800
1000
Index


Histogram of simulation_result$sigmasq
Frequency
0
10
20
30
40
1500
1550
1600
simulation_result$sigmasq


Traceplot of MCMC iteration of mu
Cummulative Estimate
31.8
32.0
32.2
32.4
0
200
400
600
800
1000
MCMC iteration



We can observe from the traceplots for mus and sigmasqs that both of them converge to a certain value, which matches the expected mean and sigmas for our original data.

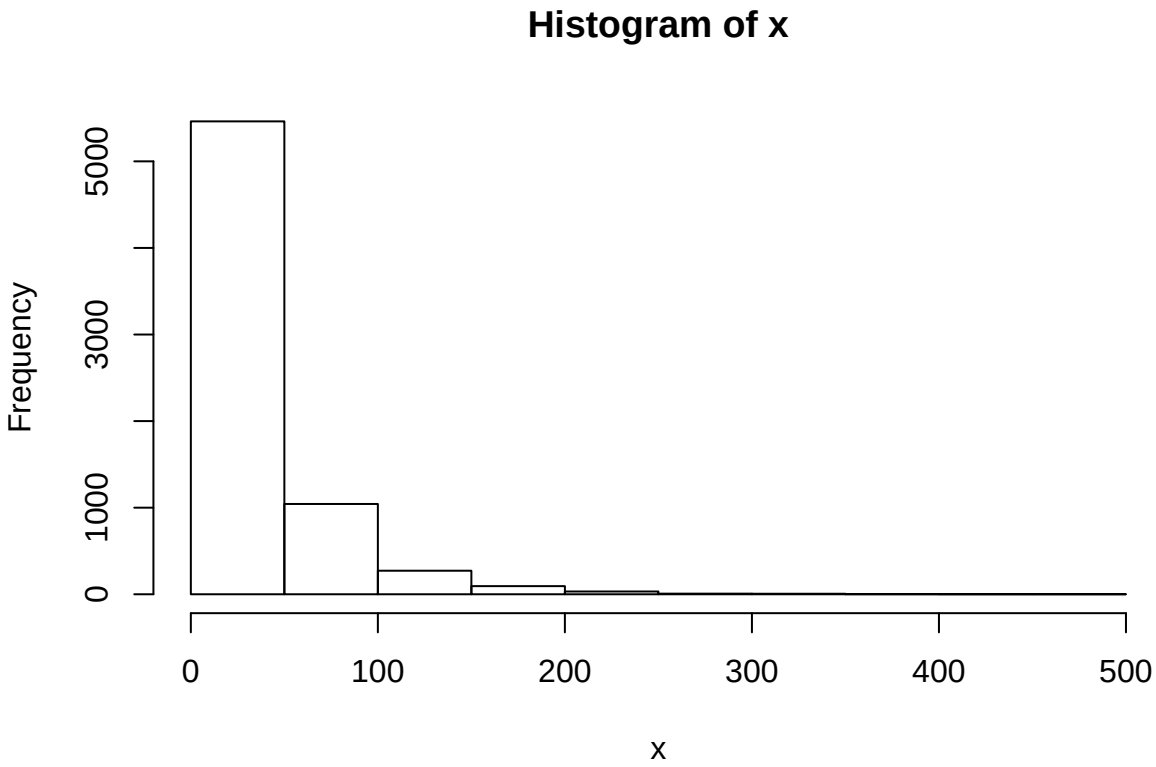
(b) Mixture normal model. Let us now instead assume that the daily precipitation $\{y_1, .., y_n\}$ follow an iid two-component mixture of normals model:

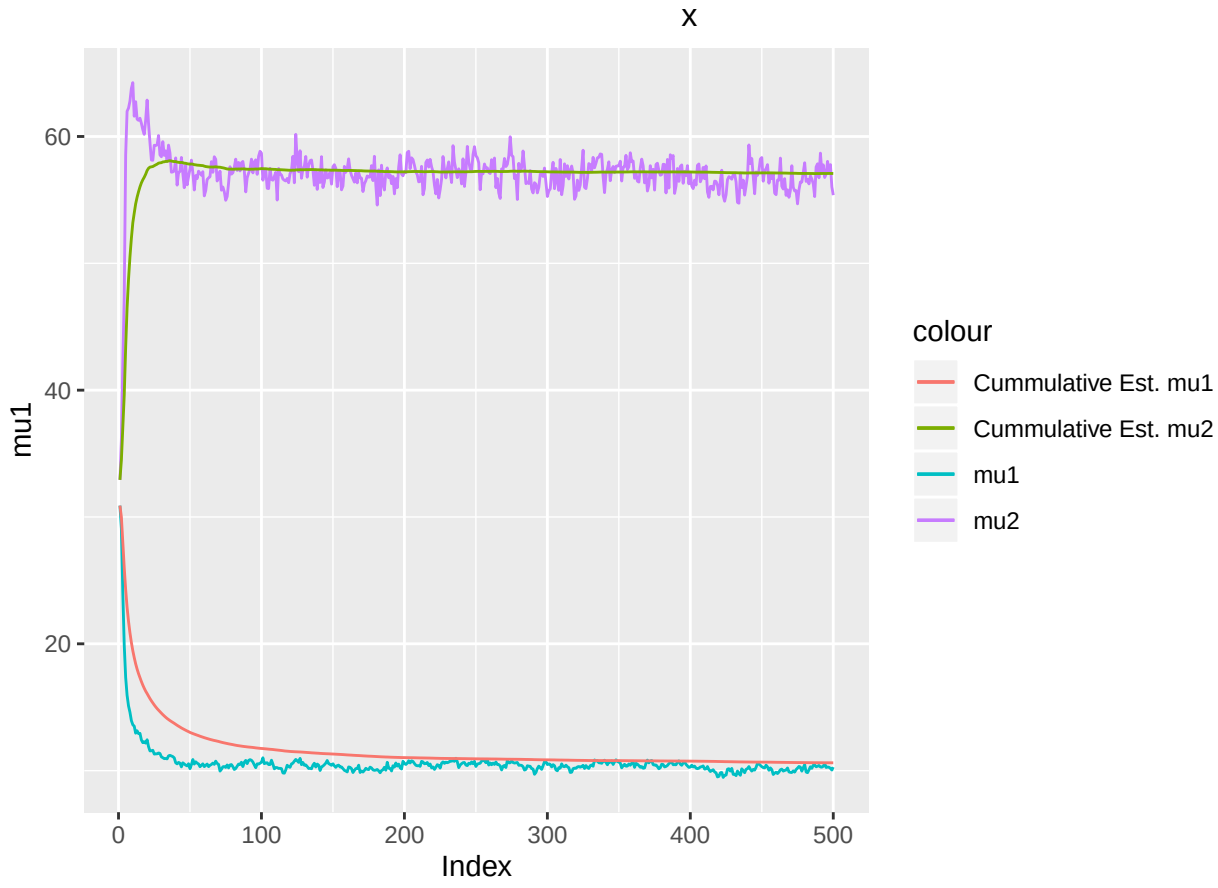
$$p(y_i|\mu, \sigma^2, \pi) = \pi N(y_i|\mu_1, \sigma_1^2) + (1 - \pi)N(y_i|\mu_2, \sigma_2^2)$$

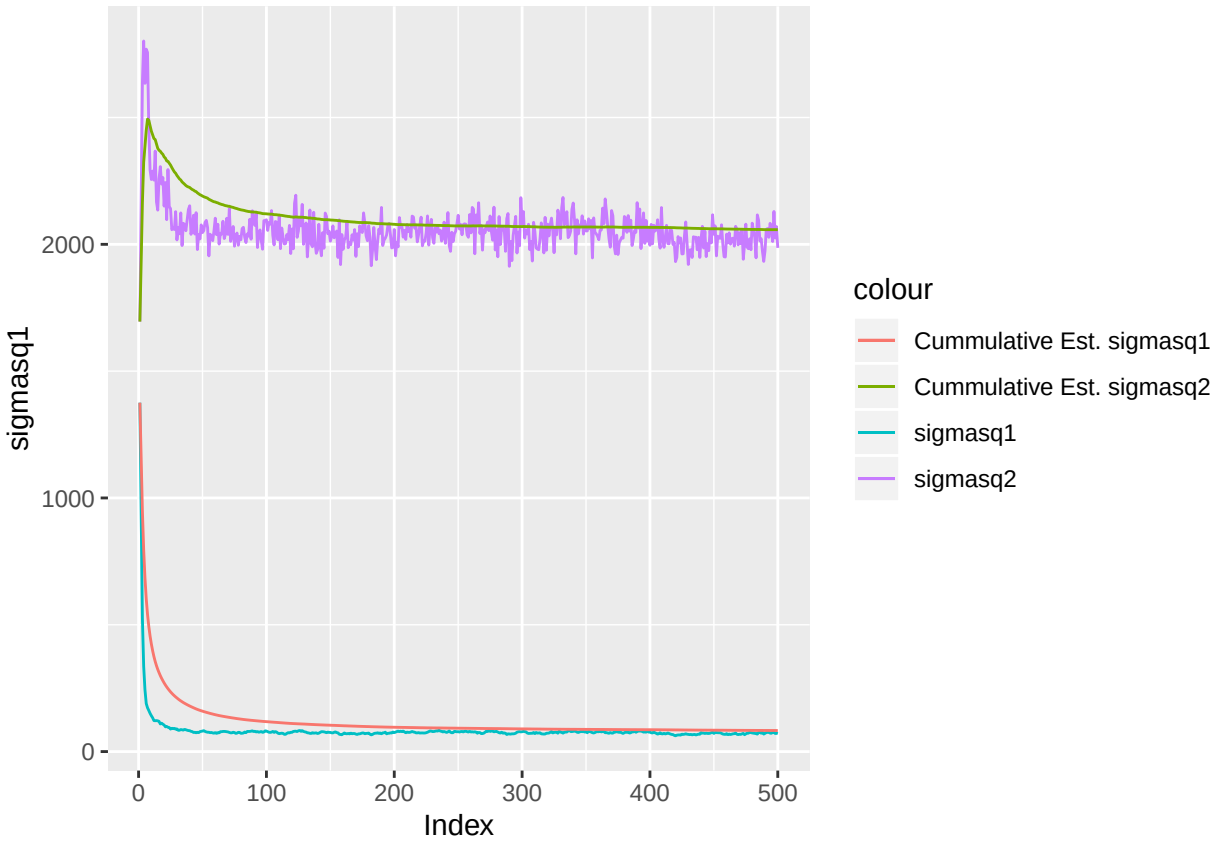
, where

$$\mu = (\mu_1, \mu_2) \; and \; \sigma^2 = (\sigma_1^2, \sigma_2^2)$$

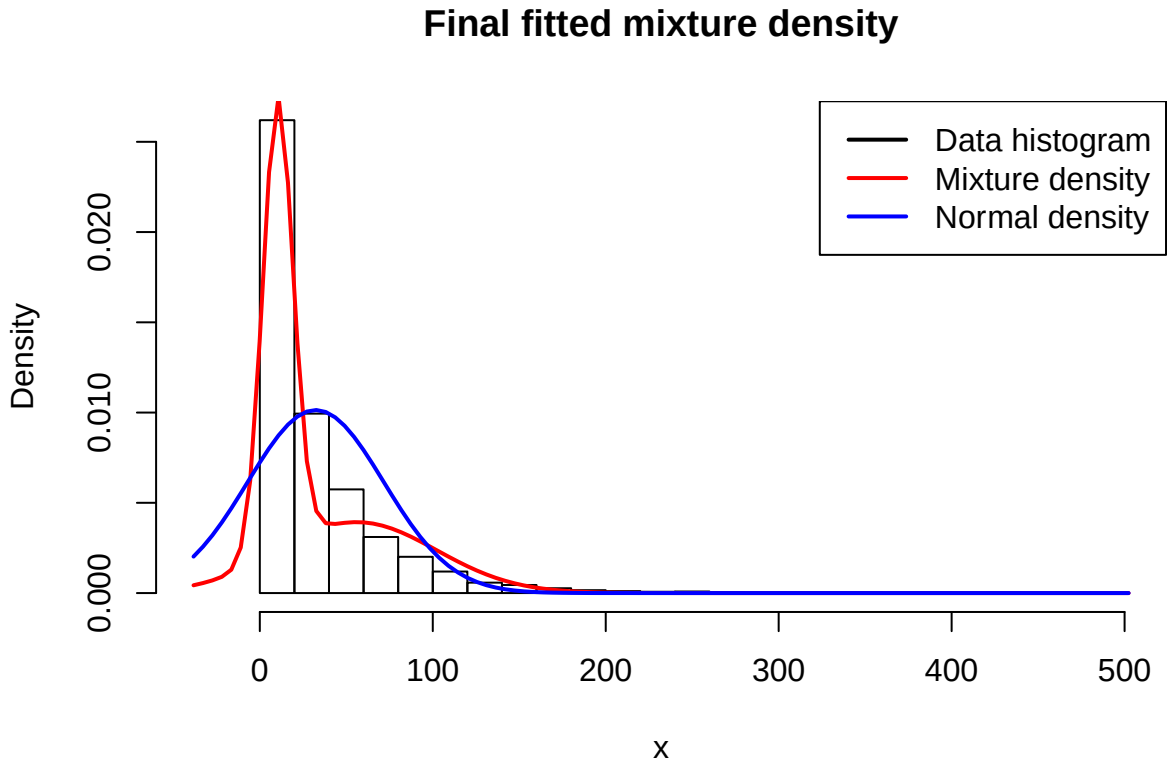
Use the Gibbs sampling data augmentation algorithm in NormalMixtureGibbs.R (available under Lecture 7 on the course page) to analyze the daily precipitation data. Set the prior hyperparameters suitably. Evaluate the convergence of the sampler.


Histogram of x
Frequency
0
1000
3000
5000
0
100
200
300
400
500
x


mu1
60
40
20
0
100
200
300
400
500
Index
colour
Cummulative Est. mu1
Cummulative Est. mu2
mu1
mu2



(c) Graphical comparison. Plot the following densities in one figure: 1) a histogram or kernel density estimate of the data. 2) Normal density $N(y_i|\mu, \sigma^2)$ in (a); 3) Mixture of normals density. $p(y_i|\mu, \sigma^2, \pi)$ in (b). Base your plots on the mean over all posterior draws.



The mixture density performed much better than normal density in this case. Although the normal density follows the same trend as our data, it reflects our data with less flexibility. But the mixture model fits better, it reflects the turning point.

2. Metropolis Random Walk for Poisson regression.

Consider the following Poisson regression model

$$y_i|\beta \sim Poisson[exp(X_i^T\beta)], i = 1, \cdots, n$$

where y_i is the count for the ith observation in the sample and xi is the p-dimensional vector with covariate observations for the ith observation. Use the data set eBayNumberOfBidderData.dat. This dataset contains observations from 1000 eBay auctions of coins. The response variable is nBids and records the number of bids in each auction. The remaining variables are features/covariates (x):

• Const (for the intercept) • PowerSeller (is the seller selling large volumes on eBay?) • VerifyID (is the seller verified by eBay?) • Sealed (was the coin sold sealed in never opened envelope?) • MinBlem (did the coin have a minor defect?) • MajBlem (a major defect?) • LargNeg (did the seller get a lot of negative feedback from customers?) • LogBook (logarithm of the coins book value according to expert sellers. Standardized) • MinBidShare (a variable that measures ratio of the minimum selling price (starting price) to the book value. Standardized).

(a) Obtain the maximum likelihood estimator of β in the Poisson regression model for the eBay data [Hint: glm.R, don't forget that glm() adds its own intercept so don't input the covariate Const]. Which covariates are significant?

```
## (Intercept) PowerSeller    VerifyID      Sealed     Minblem     MajBlem
##  1.07244206 -0.02054076 -0.39451647  0.44384257 -0.05219829 -0.22087119
##     LargNeg     LogBook MinBidShare
##  0.07067246 -0.12067761 -1.89409664
```

```
##
## Call:
## glm(formula = nBids ~ ., family = poisson(), data = data1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5800  -0.7222  -0.0441   0.5269   2.4605
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.07244    0.03077  34.848  < 2e-16 ***
## PowerSeller -0.02054    0.03678  -0.558   0.5765
## VerifyID    -0.39452    0.09243  -4.268 1.97e-05 ***
## Sealed       0.44384    0.05056   8.778  < 2e-16 ***
## Minblem     -0.05220    0.06020  -0.867   0.3859
## MajBlem     -0.22087    0.09144  -2.416   0.0157 *
## LargNeg      0.07067    0.05633   1.255   0.2096
## LogBook     -0.12068    0.02896  -4.166 3.09e-05 ***
## MinBidShare -1.89410    0.07124 -26.588  < 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: 2151.28  on 999  degrees of freedom
## Residual deviance:  867.47  on 991  degrees of freedom
```

```
## AIC: 3610.3
##
## Number of Fisher Scoring iterations: 5

## Significant covariants are:

## Constant(intercept), VerifyID, Sealed, MajBlem, LogBook, MinBidShare
```

Sealed and MinBidShare have more weight than other covatiates.

(b) Let's now do a Bayesian analysis of the Poisson regression. Let the prior be $\beta \sim N[0, 100 \cdot (X^T X)^{-1}]$ where X is the n x p covariate matrix. This is a commonly used prior which is called Zellner's g-prior. Assume first that the posterior density is approximately multivariate normal:

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

where $\tilde{\beta}$ is the posterior mode and $J_y(\tilde{\beta})$ is the negative Hessian at the posterior mode. $\tilde{\beta}$ and $J_y(\tilde{\beta})$ can be obtained by numerical optimization (optim.R) exactly like you already did for the logistic regression in Lab 2 (but with the log posterior function replaced by the corresponding one for the Poisson model, which you have to code up.).

```
##
##  Posterior Mode of betas: 1.075051 -0.02056734 -0.3960252 0.4441226 -0.05193492 -0.2205022 0.0706384
##
##  Posterior Covariance:
```

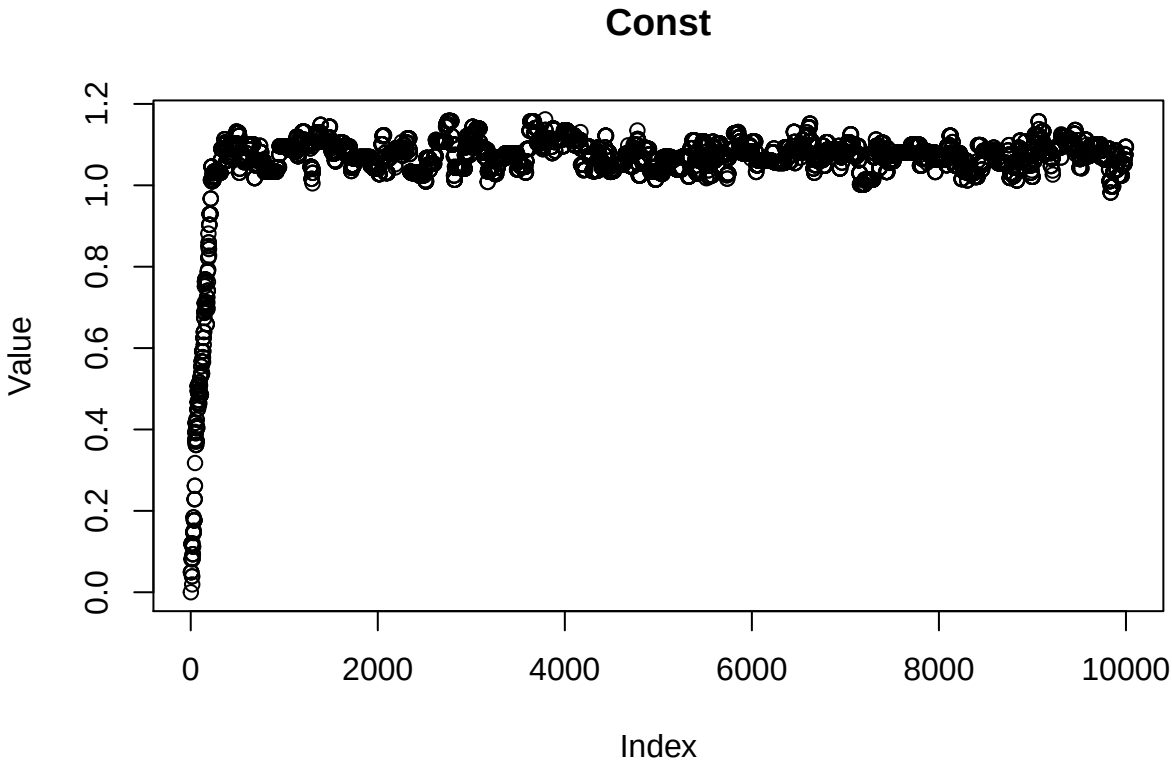
0.0009487	-0.0007137	-0.0002720	-0.0002720	-0.0004458	-0.0002768	-0.0005129	0.0000663	0.0011233
-0.0007137	0.0013528	0.0000436	-0.0002958	0.0001142	-0.0002090	0.0002807	0.0001182	-0.0005683
-0.0002720	0.0000436	0.0085699	-0.0007901	-0.0001035	0.0002267	0.0003308	-0.0003150	-0.0004152
-0.0002720	-0.0002958	-0.0007901	0.0025554	0.0003580	0.0004535	0.0003374	-0.0001326	-0.0000645
-0.0004458	0.0001142	-0.0001035	0.0003580	0.0036237	0.0003491	0.0000581	0.0000581	-0.0000667
-0.0002768	-0.0002090	0.0002267	0.0004535	0.0003491	0.0083560	0.0004042	-0.0000899	0.0002604
-0.0005129	0.0002807	0.0003308	0.0003374	0.0000581	0.0004042	0.0031715	-0.0002545	-0.0001076
0.0000663	0.0001182	-0.0003150	-0.0001326	0.0000581	-0.0000899	-0.0002545	0.0008394	0.0010436
0.0011233	-0.0005683	-0.0004152	-0.0000645	-0.0000667	0.0002604	-0.0001076	0.0010436	0.0050956

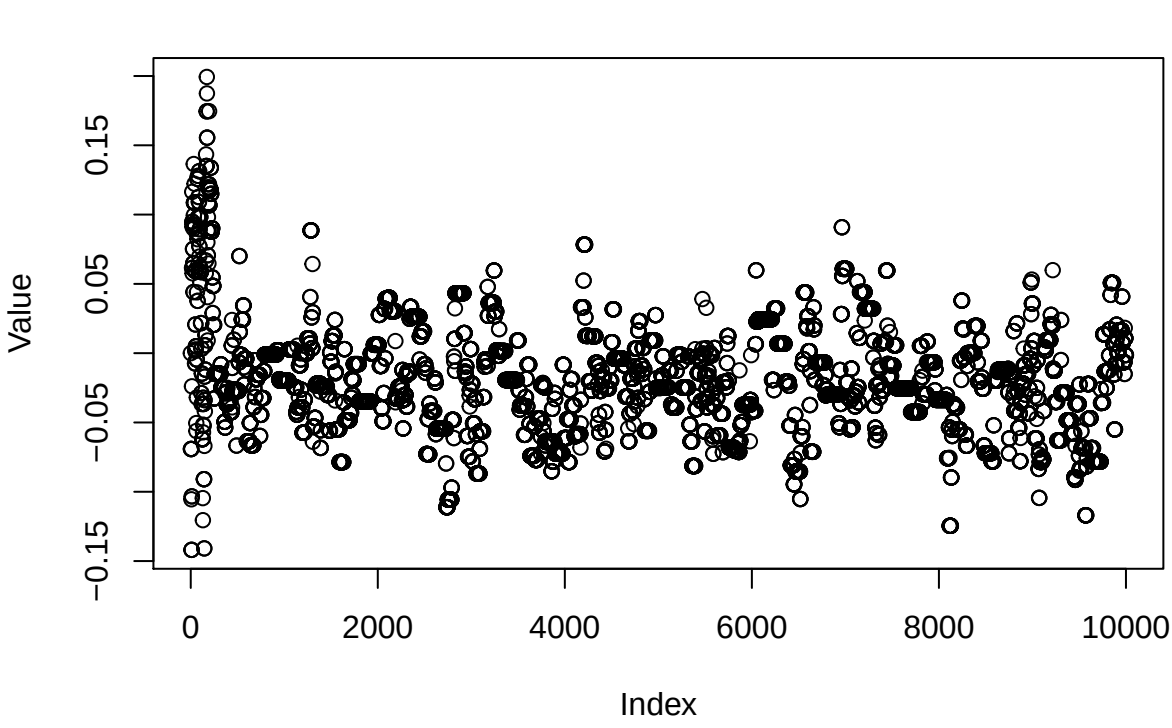
(c) Now, let's simulate from the actual posterior of β using the Metropolis algo- rithm and compare with the approximate results in b). Program a general function that uses the Metropolis algorithm to generate random draws from an arbitrary posterior density. In order to show that it is a general function for any model, I will denote the vector of model parameters by θ. Let the proposal density be the multivariate normal density mentioned in Lecture 8 (random walk Metropolis).

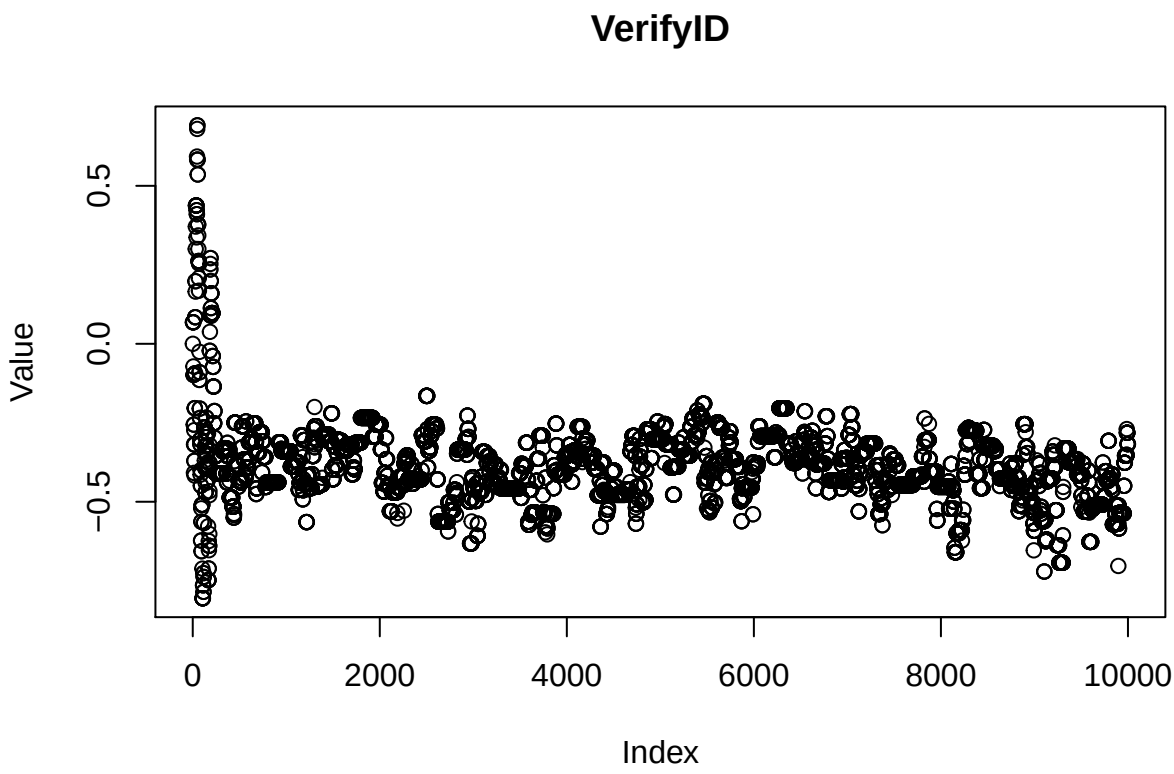
$$\theta_p|\theta^{(i-1)} \sim N(\theta^{(i-1)}, c\Sigma)$$

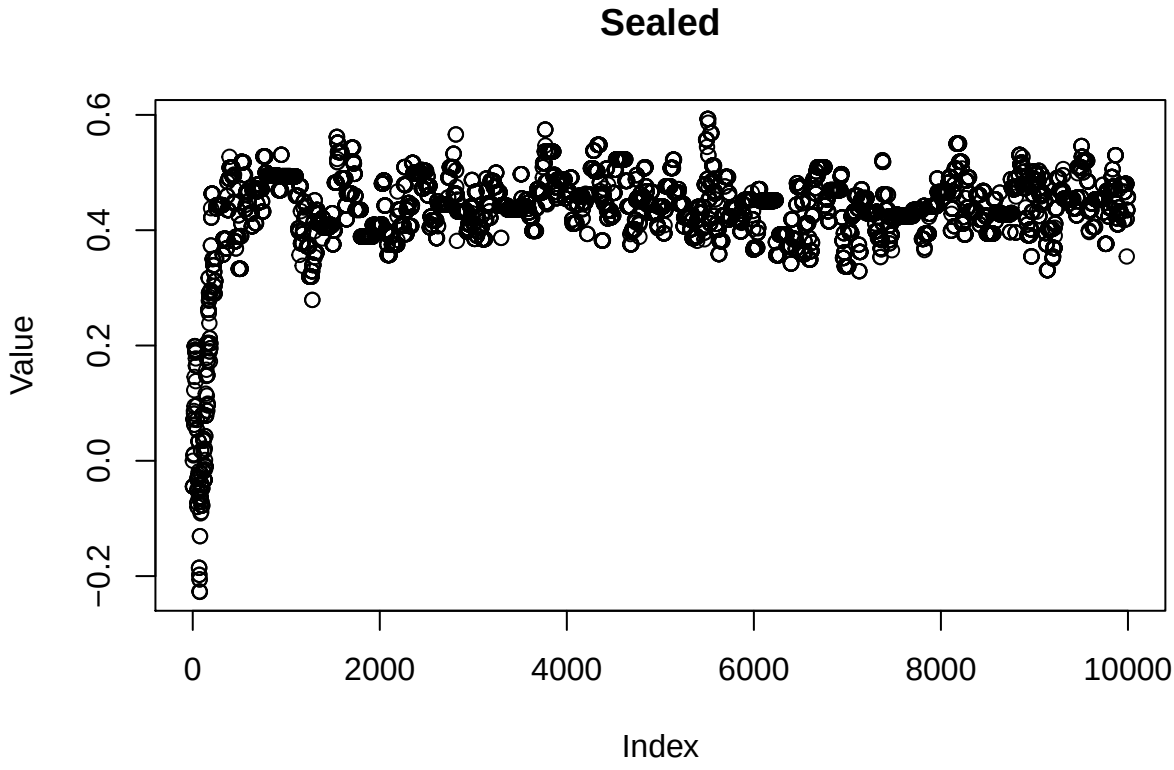
where $\sigma = J_y^{-1}(\tilde{\beta})$ obtained in b). The value c is a tuning parameter and should be an input to your Metropolis function. The user of your Metropo- lis function should be able to supply her own posterior density function, not necessarily for the Poisson regression, and still be able to use your Metropolis function. This is not so straightforward, unless you have come across function objects in R and the triple dot (. . .) wildcard argument. I have posted a note (HowToCodeRWM.pdf) on the course web page that describes how to do this in R.

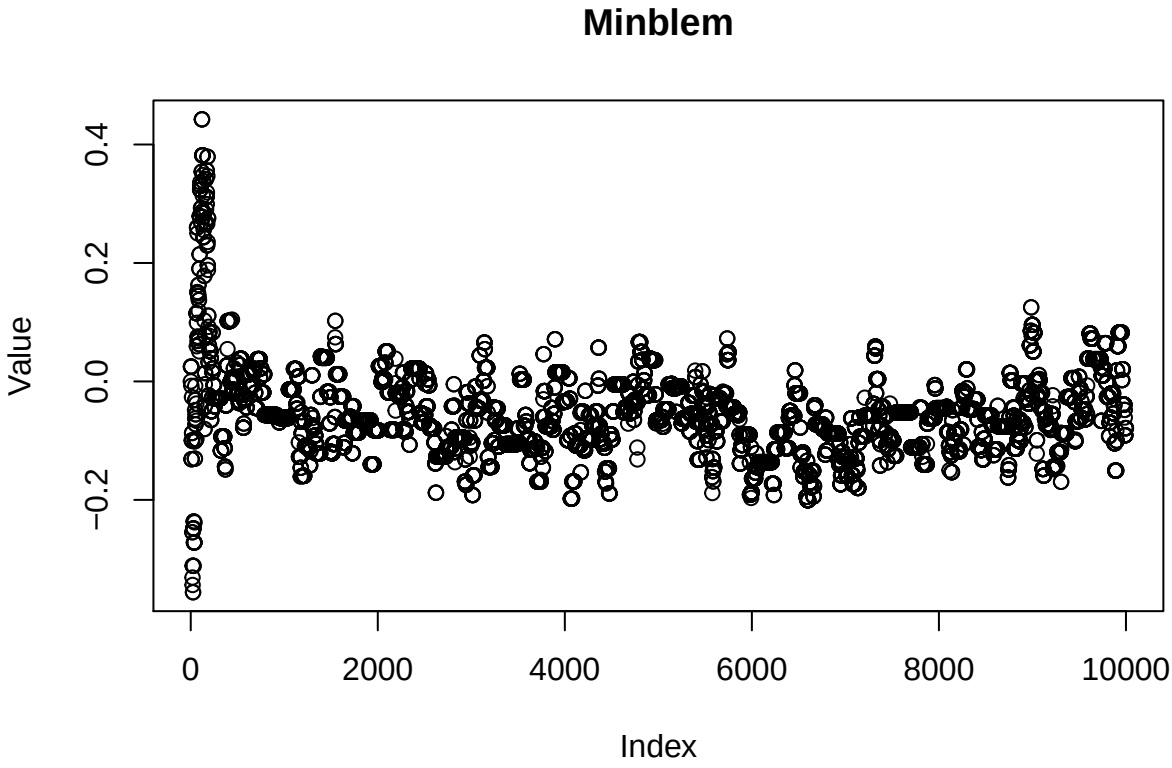
Now, use your new Metropolis function to sample from the posterior of β in the Poisson regression for the eBay dataset. Assess MCMC convergence by graphical methods.

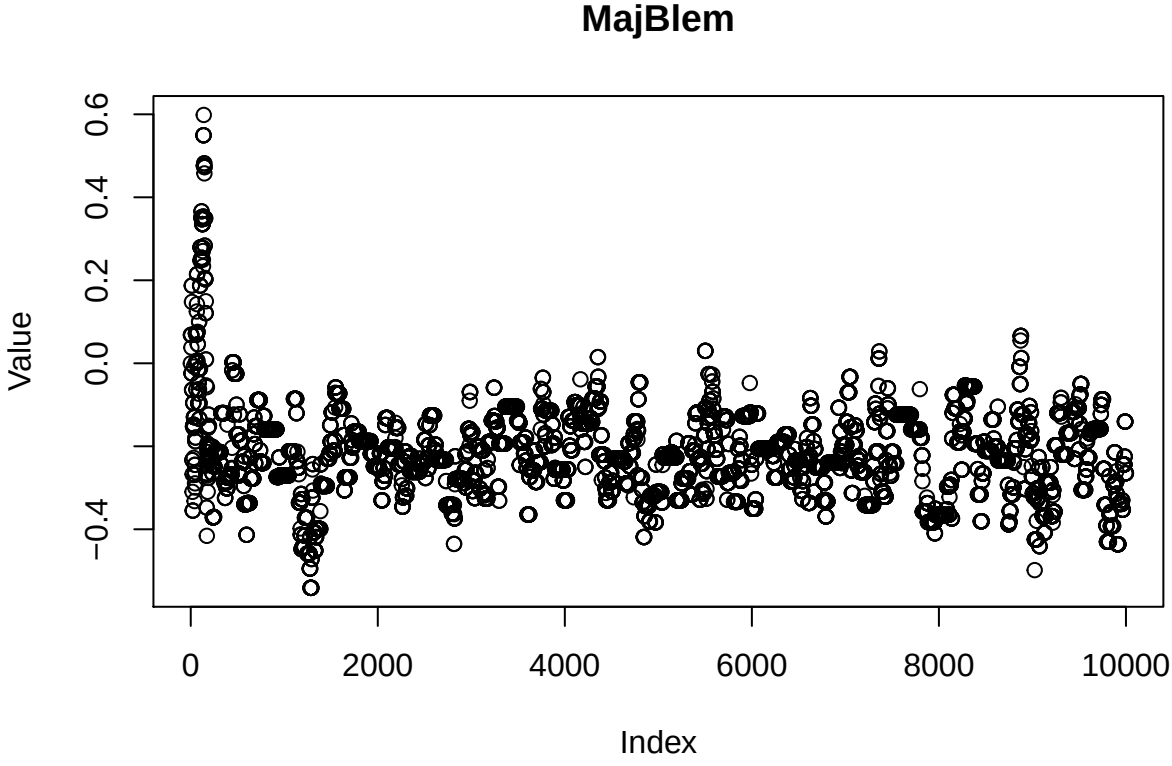

Const
Value
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
2000
4000
6000
8000
10000
Index

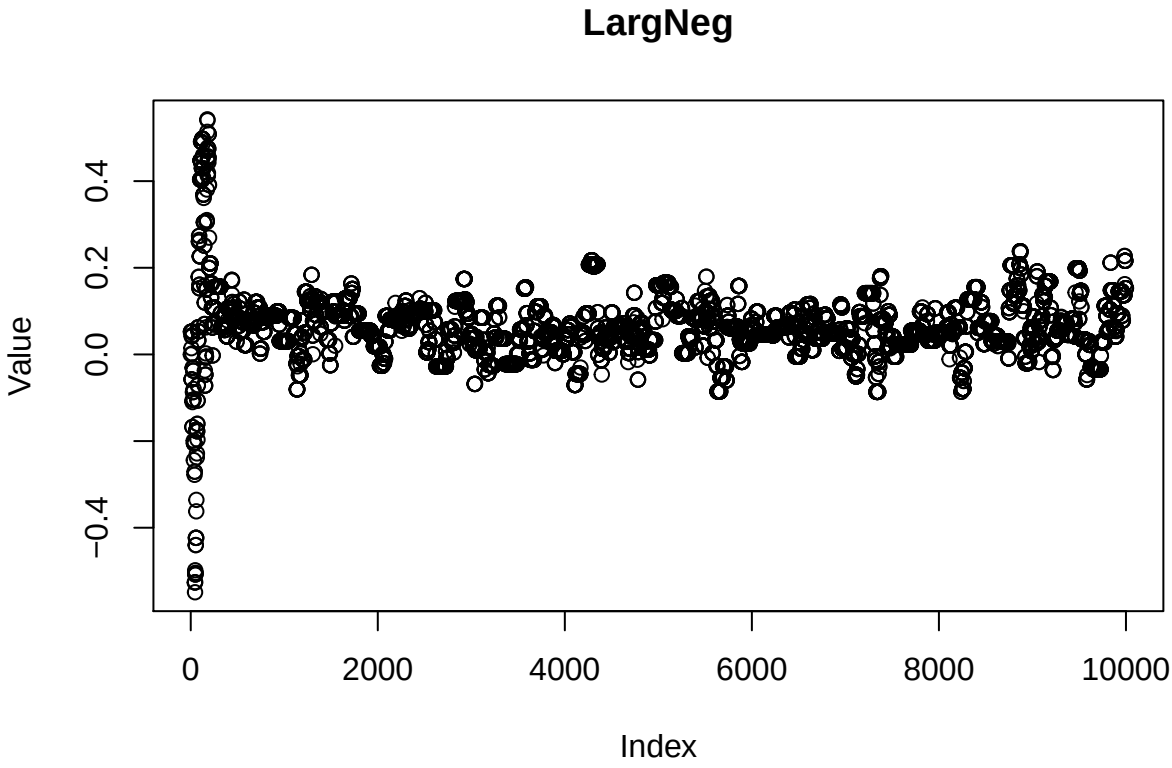

PowerSeller
Value
−0.15
−0.05
0.05
0.15
0
2000
4000
6000
8000
10000
Index

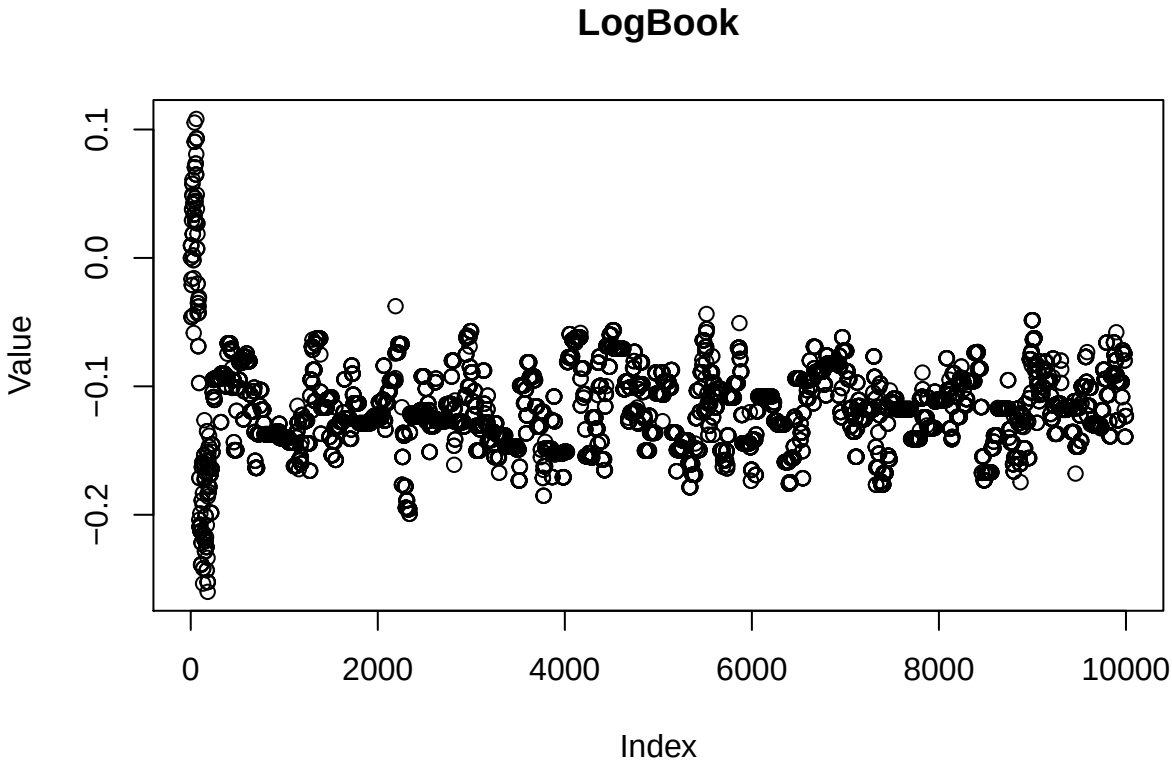

VerifyID
Value
0.5
0.0
−0.5
0
2000
4000
6000
8000
10000
Index

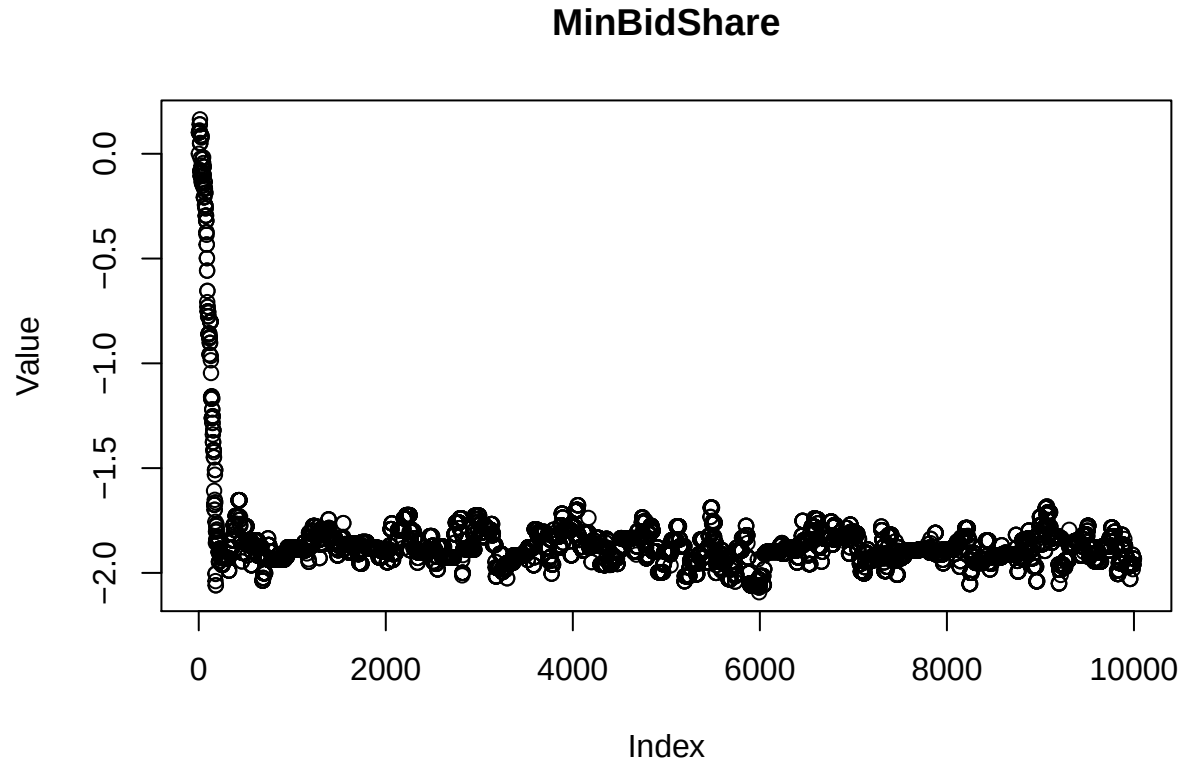

Sealed
Value
0.6
0.4
0.2
0.0
−0.2
0
2000
4000
6000
8000
10000
Index

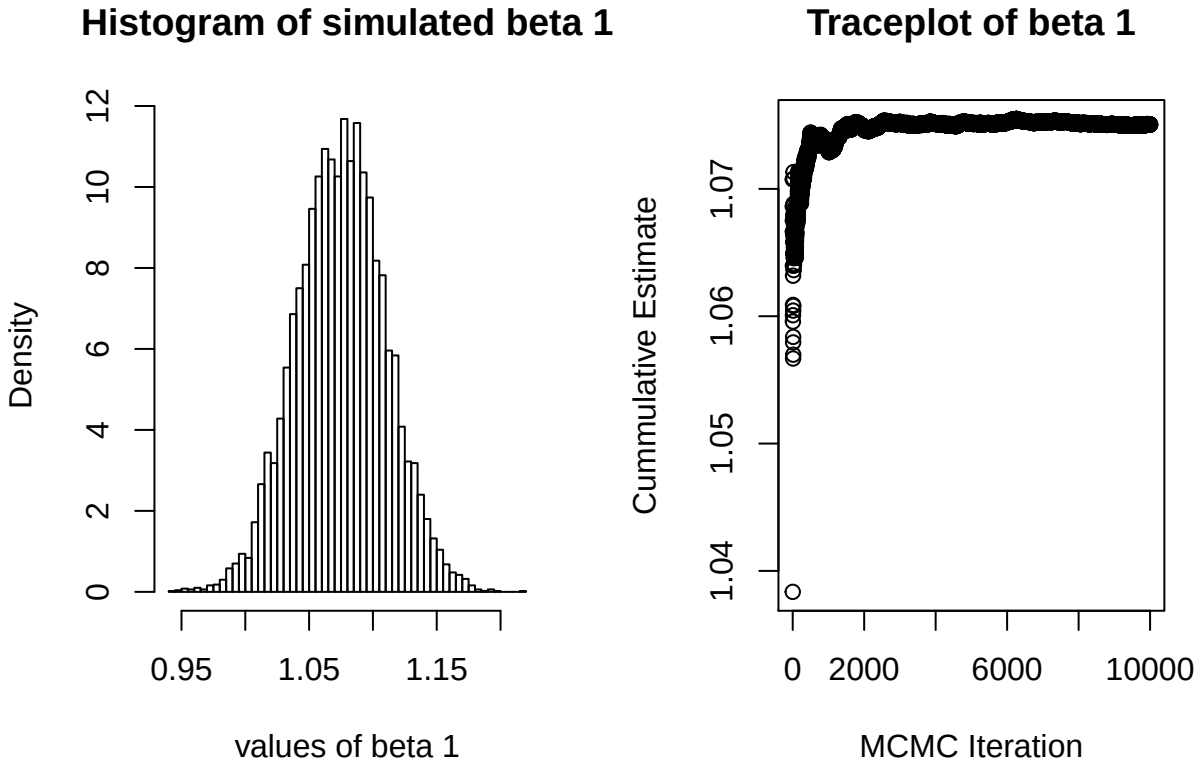

Minblem
Value
0.4
0.2
0.0
−0.2
0
2000
4000
6000
8000
10000
Index

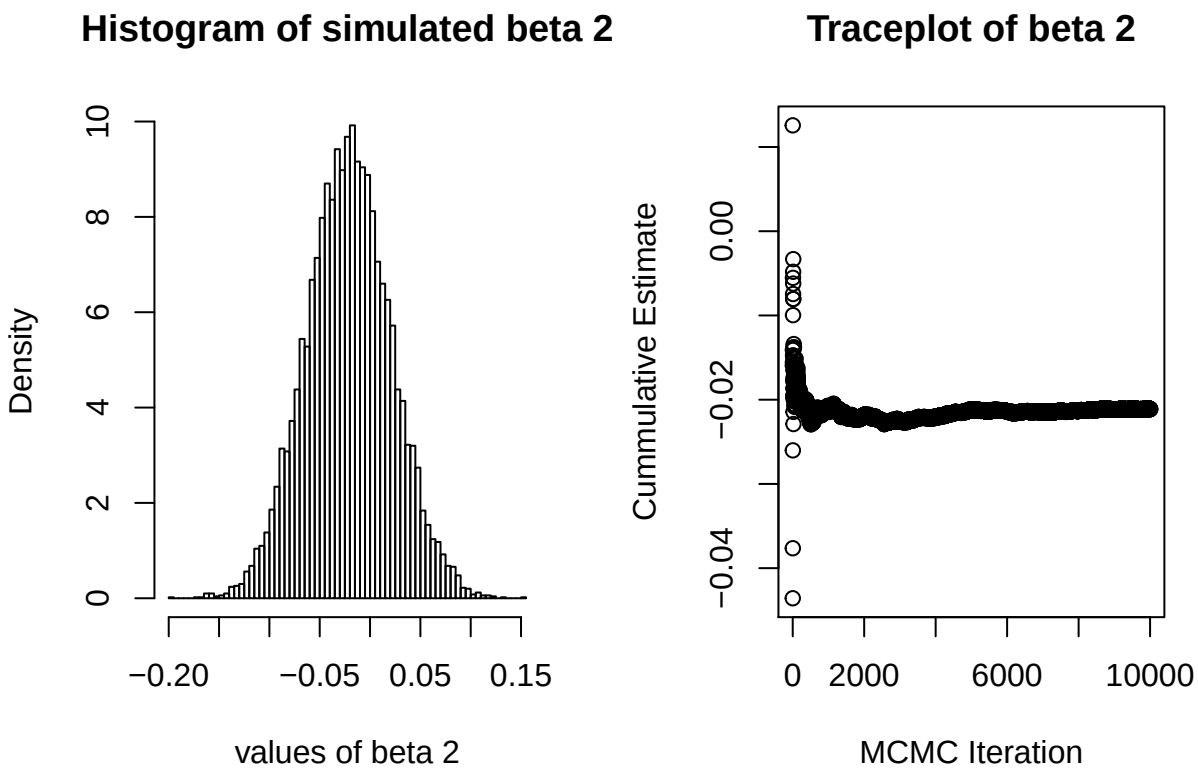

MajBlem
Value
0.6
0.4
0.2
0.0
−0.4
0
2000
4000
6000
8000
10000
Index

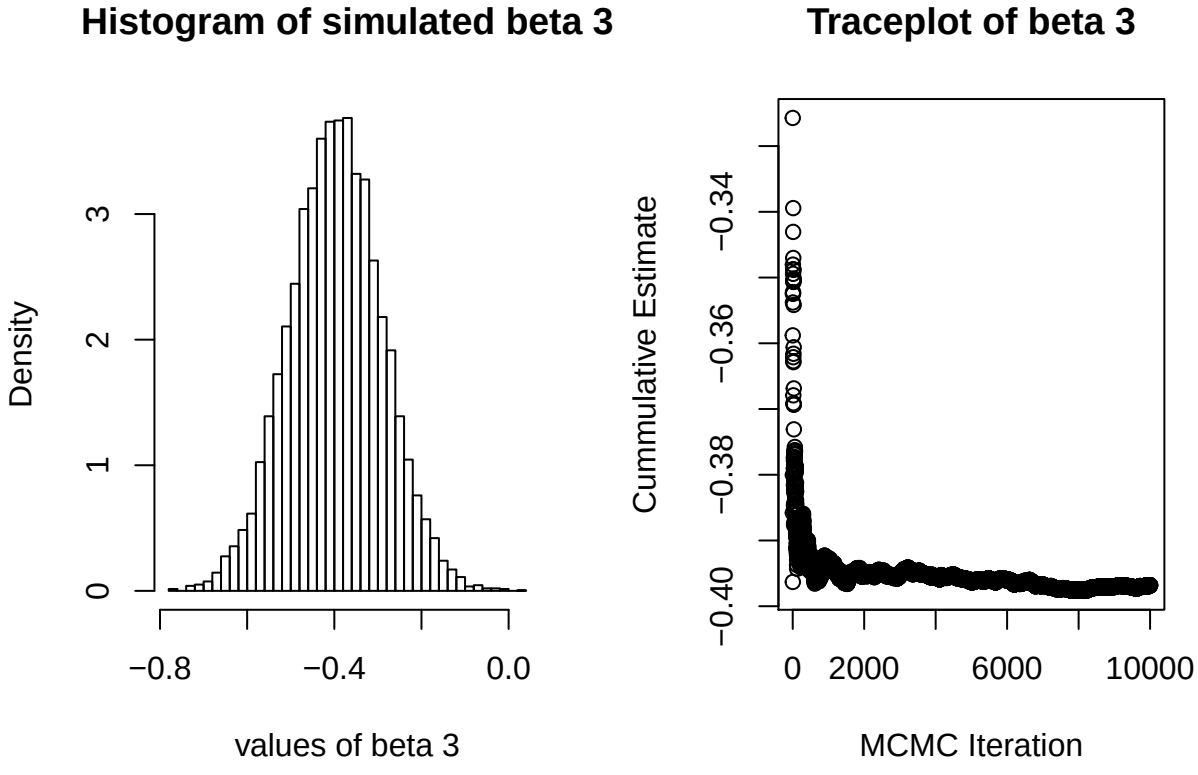

LargNeg
Value
0.4
0.2
0.0
−0.4
0
2000
4000
6000
8000
10000
Index

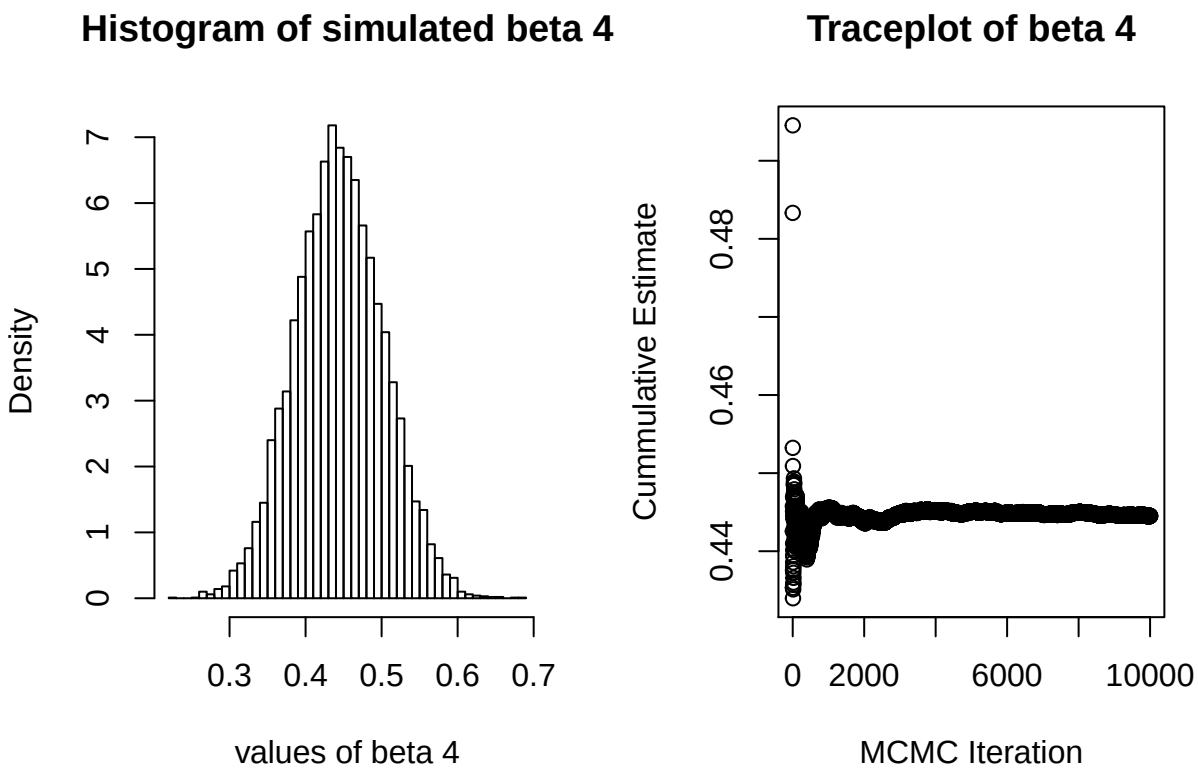

LogBook
Value
0.1
0.0
−0.1
−0.2
0
2000
4000
6000
8000
10000
Index

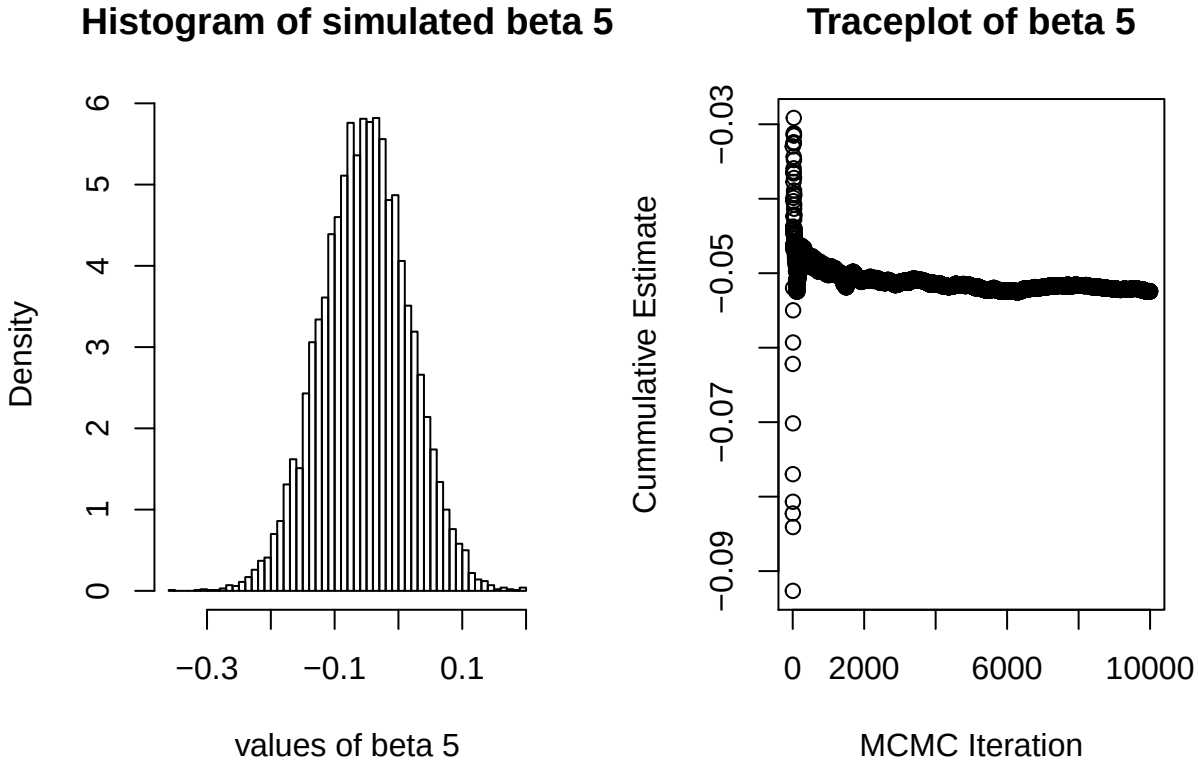

MinBidShare
Value
−2.0
−1.5
−1.0
−0.5
0.0
0
2000
4000
6000
8000
10000
Index

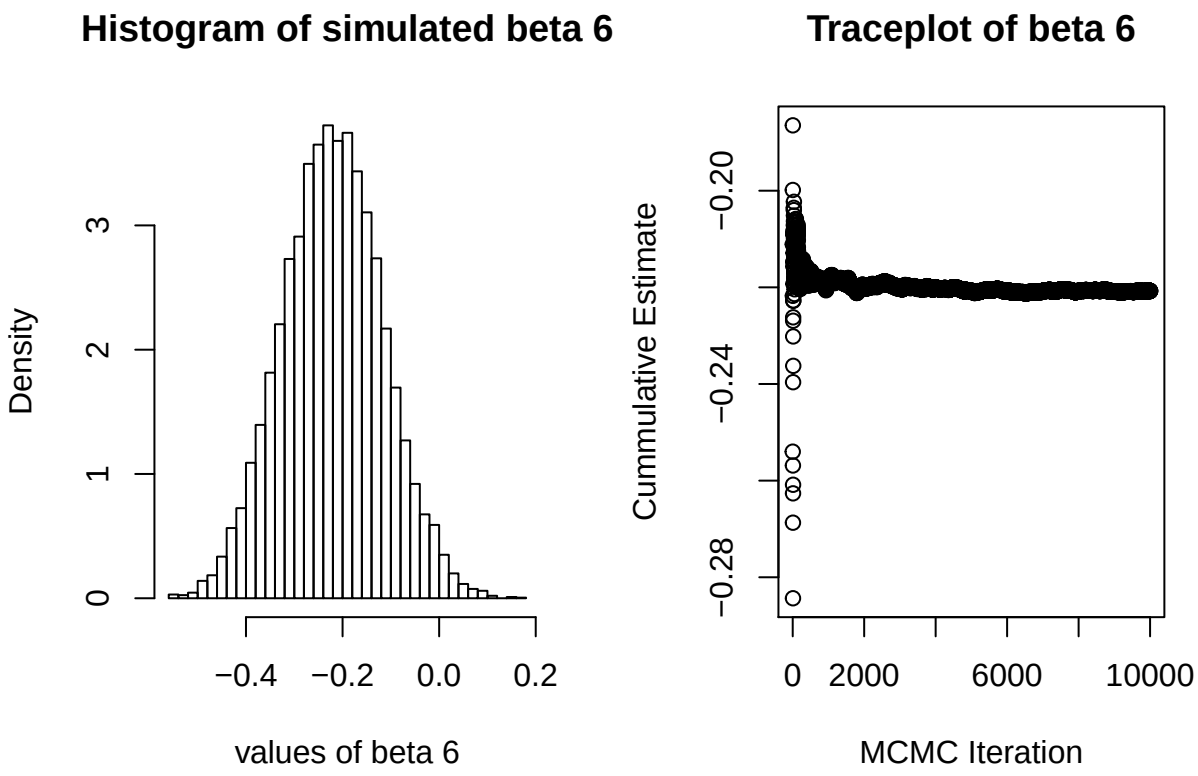

Histogram of simulated beta 1
Density
0
2
4
6
8
10
12
0.95
1.05
1.15
values of beta 1
Traceplot of beta 1
Cummulative Estimate
1.04
1.05
1.06
1.07
0
2000
6000
10000
MCMC Iteration

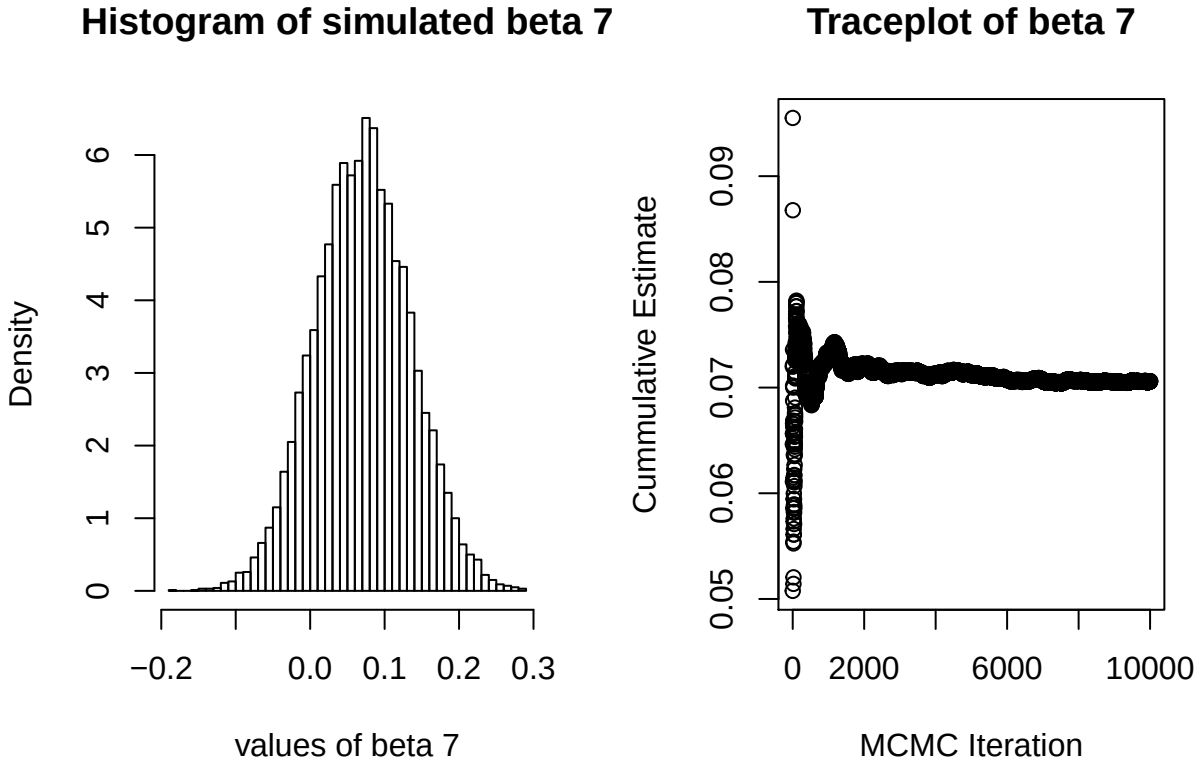

Histogram of simulated beta 2
Density
0
2
4
6
8
10
−0.20
−0.05
0.05
0.15
values of beta 2
Traceplot of beta 2
Cummulative Estimate
−0.04
−0.02
0.00
0
2000
6000
10000
MCMC Iteration

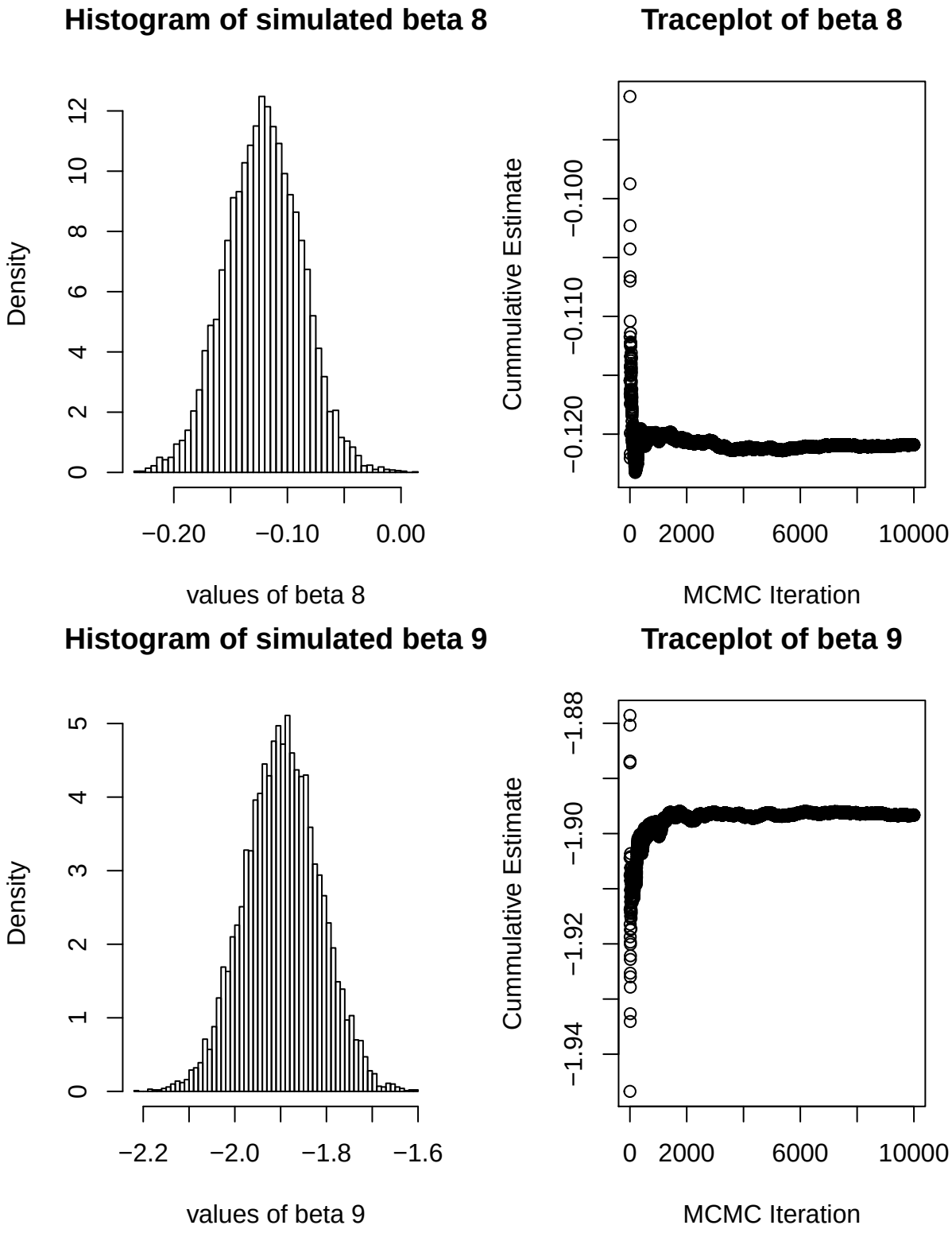

Histogram of simulated beta 3
Density
0
1
2
3
−0.8
−0.4
0.0
values of beta 3
Traceplot of beta 3
Cummulative Estimate
−0.40
−0.38
−0.36
−0.34
0
2000
6000
10000
MCMC Iteration


Histogram of simulated beta 4
Density
0
1
2
3
4
5
6
7
0.3
0.4
0.5
0.6
0.7
values of beta 4
Traceplot of beta 4
Cummulative Estimate
0.44
0.46
0.48
0
2000
6000
10000
MCMC Iteration


Histogram of simulated beta 5
Density
0
1
2
3
4
5
6
−0.3
−0.1
0.1
values of beta 5
Traceplot of beta 5
Cummulative Estimate
−0.09
−0.07
−0.05
−0.03
0
2000
6000
10000
MCMC Iteration


Histogram of simulated beta 6
Density
0
1
2
3
−0.4
−0.2
0.0
0.2
values of beta 6
Traceplot of beta 6
Cummulative Estimate
−0.20
−0.24
−0.28
0
2000
6000
10000
MCMC Iteration

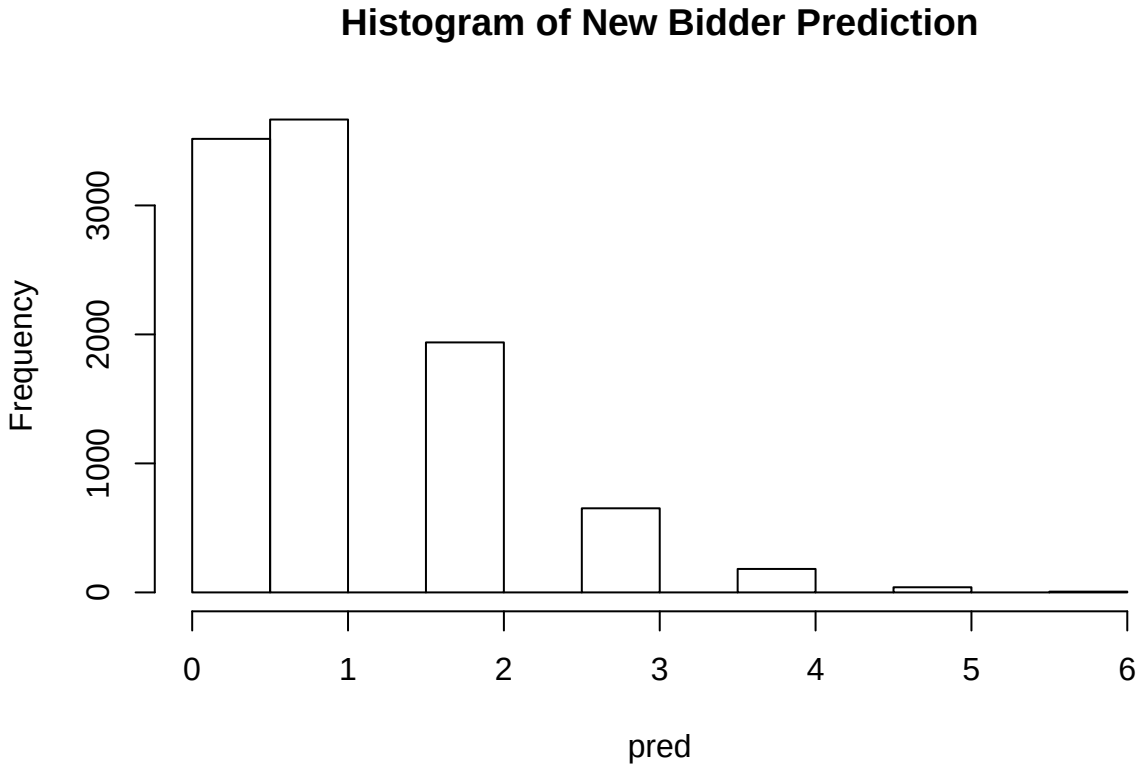

Histogram of simulated beta 7
Density
0
1
2
3
4
5
6
−0.2
0.0
0.1
0.2
0.3
values of beta 7
Traceplot of beta 7
Cummulative Estimate
0.05
0.06
0.07
0.08
0.09
0
2000
6000
10000
MCMC Iteration



(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?

• PowerSeller = 1 • VerifyID = 1 • Sealed = 1 • MinBlem = 0 • MajBlem = 0 • LargNeg = 0 • LogBook

$= 1$ • MinBidShare $= 0.5$




```
## Hte probability of no bidder in this new auction is  0.3516
```

Appendix A : Code Question 1

```
data1 = read.table("rainfall.dat", header = FALSE)
#1-1. Normal Model

## defining function for simulation of mu and sigmasq
library(extraDistr)
joint_simulation = function(data, niteration){


  ### initial nu is set to be 1. tau value is from the data.
  ### when scale parameter is fixed, the tail of the distribution is heavier
  ### as the degrees of freedom(nu) is smaller. Since the true sigmasq is
  ### unknown, distribution with heavy tail is used to represent its uncertainty
  sigmasq = rinvchisq(1, nu=1, tau= var(data[,1]))
  init_mu = mean(data[,1])
  init_sigmasq = var(data[,1])

  n = dim(data)[1]

  result_store = data.frame(mu=init_mu, sigmasq = init_sigmasq)


  for (i in 1:niteration){
    w = (n/sigmasq) / ( (n/sigmasq) + (1/init_sigmasq) )
    tausq = 1 / ( (n/sigmasq) + (1/init_sigmasq) )
    new_mu = rnorm(1, mean = (w*(mean(data[,1]))) + ((1-w)*init_mu), sd = sqrt(tausq) )
    scaling = sum((data[,1] - new_mu)^2) / (n-1)
    new_sigmasq = rinvchisq(1, nu = n-1, tau = scaling)
    new = data.frame(mu = new_mu, sigmasq = new_sigmasq)
    result_store = rbind(result_store, new)
    init_mu = new_mu
    init_sigmasq = result_store$sigmasq[i]

  }

  return(result_store)
}

### simulation and plotting
set.seed(12345)
simulation_result = joint_simulation(data = data1, niteration = 1000)
### raw data plotting
plot(simulation_result$mu,ylab = "Simulations for mus")
hist(simulation_result$mu,breaks = 100,)
plot(simulation_result$sigmasq,ylab = "Simulations for sigmasqs")
hist(simulation_result$sigmasq, breaks = 100)
cum_mu = cumsum(simulation_result$mu)
for (i in 1:length(cum_mu)){
  cum_mu[i] = cum_mu[i]/i
}
plot(cum_mu, main= "Traceplot of MCMC iteration of mu", xlab="MCMC iteration", ylab="Cummulative Estima
```

```r
cum_sigmasq = cumsum(simulation_result$sigmasq)
for (i in 1:length(cum_sigmasq)) {
  cum_sigmasq[i] = cum_sigmasq[i]/i
}
plot(cum_sigmasq, main= "Traceplot of MCMC iteration of sigma^2", xlab="MCMC iteration", ylab="Cummulati
x <- as.matrix(data1[,1])

# Model options
nComp <- 2    # Number of mixture components

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- c(20, 160) # Prior mean of mu by seeing the histogram of the data
tau2Prior <- rep(10,nComp) # Prior std of mu
sigma2_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)
nu0 <- rep(3,nComp) # degrees of freedom for prior on sigma2, set it as small value to make prior distr

# MCMC options
nIter <- 500 # Number of Gibbs sampling draws

# Plotting options
plotFit <- TRUE
lineColors <- c("blue", "green")
sleepTime <- 0 # Adding sleep time between iterations for plotting
################   END USER INPUT ################

###### Defining a function that simulates from the
rScaledInvChi2 <- function(n, df, scale){
  return((df*scale)/rchisq(n,df=df))
}

####### Defining a function that simulates from a Dirichlet distribution
rDirichlet <- function(param){
  nCat <- length(param)
  piDraws <- matrix(NA,nCat,1)
  for (j in 1:nCat){
    piDraws[j] <- rgamma(1,param[j],1)
  }
  piDraws = piDraws/sum(piDraws) # Diving every column of piDraws by the sum of the elements in that co
  return(piDraws)
}

# Simple function that converts between two different representations of the mixture allocation
S2alloc <- function(S){
  n <- dim(S)[1]
  alloc <- rep(0,n)
  for (i in 1:n){
    alloc[i] <- which(S[i,] == 1)
  }
  return(alloc)
}

# Initial value for the MCMC
```

```r
nObs <- length(x)
S <- t(rmultinom(nObs, size = 1 , prob = rep(1/nComp,nComp))) # nObs-by-nComp matrix with component all
mu <- quantile(x, probs = seq(0,1,length = nComp))
sigma2 <- rep(var(x),nComp)
probObsInComp <- rep(NA, nComp)

# Setting up the plot
xGrid <- seq(min(x)-1*apply(x,2,sd),max(x)+1*apply(x,2,sd),length = 100)
xGridMin <- min(xGrid)
xGridMax <- max(xGrid)
mixDensMean <- rep(0,length(xGrid))
effIterCount <- 0
ylim <- c(0,2*max(hist(x)$density))

storage_mu = data.frame(mu1 = as.numeric(), mu2 = as.numeric(),stringsAsFactors = FALSE)
storage_sigmasq = data.frame( sigmasq1 = as.numeric(), sigmasq2 = as.numeric(), stringsAsFactors = FALSE

for (k in 1:nIter){
  #message(paste('Iteration number:',k))
  alloc <- S2alloc(S) # Just a function that converts between different representations of the group al
  nAlloc <- colSums(S)
  #print(nAlloc)
  # Update components probabilities
  pi <- rDirichlet(alpha + nAlloc)

  # Update mu's
  for (j in 1:nComp){
    precPrior <- 1/tau2Prior[j]
    precData <- nAlloc[j]/sigma2[j]
    precPost <- precPrior + precData
    wPrior <- precPrior/precPost
    muPost <- wPrior*muPrior + (1-wPrior)*mean(x[alloc == j])
    tau2Post <- 1/precPost
    mu[j] <- rnorm(1, mean = muPost, sd = sqrt(tau2Post))
  }
  new_mu = data.frame(mu1 = mu[1], mu2 = mu[2])
  storage_mu = rbind(storage_mu, new_mu)

  # Update sigma2's
  for (j in 1:nComp){
    sigma2[j] <- rScaledInvChi2(1, df = nu0[j] + nAlloc[j], scale = (nu0[j]*sigma2_0[j] + sum((x[alloc =
  }
  new_sigma2 = data.frame(sigmasq1 = sigma2[1], sigmasq2 = sigma2[2])
  storage_sigmasq = rbind(storage_sigmasq, new_sigma2)


  # Update allocation
  for (i in 1:nObs){
    for (j in 1:nComp){
      probObsInComp[j] <- pi[j]*dnorm(x[i], mean = mu[j], sd = sqrt(sigma2[j]))
    }
    S[i,] <- t(rmultinom(1, size = 1 , prob = probObsInComp/sum(probObsInComp)))
  }
```

```
  # Printing the fitted density against data histogram
  if (plotFit && (k%%1 ==0)){
    effIterCount <- effIterCount + 1
    #hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = paste("Iteration number",k)
    mixDens <- rep(0,length(xGrid))
    components <- c()
    for (j in 1:nComp){
      compDens <- dnorm(xGrid,mu[j],sd = sqrt(sigma2[j]))
      mixDens <- mixDens + pi[j]*compDens
      #lines(xGrid, compDens, type = "l", lwd = 2, col = lineColors[j])
      components[j] <- paste("Component ",j)
    }
    mixDensMean <- ((effIterCount-1)*mixDensMean + mixDens)/effIterCount

    #lines(xGrid, mixDens, type = "l", lty = 2, lwd = 3, col = 'red')
    #legend("topleft", box.lty = 1, legend = c("Data histogram",components, 'Mixture'),
     #       col = c("black",lineColors[1:nComp], 'red'), lwd = 2)
    Sys.sleep(sleepTime)
  }

}
cum_mu1b = cumsum(storage_mu$mu1)
cum_mu2b = cumsum(storage_mu$mu2)
cum_sigmasq1b = cumsum(storage_sigmasq$sigmasq1)
cum_sigmasq2b = cumsum(storage_sigmasq$sigmasq2)

for (i in 1:length(cum_mu1b)){
  cum_mu1b[i] = cum_mu1b[i]/i
  cum_mu2b[i] = cum_mu2b[i]/i
  cum_sigmasq1b[i] = cum_sigmasq1b[i]/i
  cum_sigmasq2b[i] = cum_sigmasq2b[i]/i
}

ggplot(data=storage_mu)+geom_line(aes(x=seq(1:nIter),y=mu1,colour="mu1"))+geom_line(aes(x=seq(1:nIter),
ggplot(data=storage_sigmasq)+geom_line(aes(x=seq(1:nIter),y=sigmasq1,colour="sigmasq1"))+geom_line(aes(
mu_normal=mean(simulation_result$mu)
sigma_normal=mean(simulation_result$sigmasq)
hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = "Final fitted mixture density")
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 1, col = "red")
lines(xGrid, dnorm(xGrid, mean = mu_normal, sd = sqrt(sigma_normal)), type = "l", lwd = 2, col = "blue"
legend("topright", box.lty = 1, legend = c("Data histogram","Mixture density", "Normal density"), col=c
```

Appendix B : Code Question 2

```
rm(list = ls())
data=read.table("eBayNumberOfBidderData.dat",header = TRUE)
data1=data[,-2]
glm_mod=glm(nBids~.,data=data1,family=poisson())
# or it is possible to code above as:
# glm_mod = glm(nBids~0+., data = data, family = "poisson")
glm_mod$coefficients
summary(glm_mod)
cat("Significant covariants are: ")
cat("Constant(intercept), VerifyID, Sealed, MajBlem, LogBook, MinBidShare", "\n")
response=data[,1]
covariates=data[,-1]
covariates=as.matrix(covariates)
response=as.matrix(response)
p=dim(covariates)[2]
n=dim(covariates)[1]

log_likehood_poisson=function(y,x,betas){
  log_likelihood=vector(length = n)
  for(i in 1:n){
    log_likelihood[i]=y[i]*betas%*%x[i,]-exp(betas%*%x[i,])-log(factorial(y[i]))
  }
  return(sum(log_likelihood))
}

log_prior_likelihood=function(betas){
  mu=rep(0,p)
  sigma_sq=100*solve(t(covariates)%*%covariates)
  log_likelihood=(log(det(sigma_sq))+t(betas-mu)%*%solve(sigma_sq)%*%(betas-mu)+p*log(2*pi))*0.5
  return(log_likelihood)
}

# for log posterior= log prior +log likelihood
log_posterior=function(betas,y,x){
  #return(log_likelihood_probit(y,x,betas)+log_prior_likelihood(tau,betas))
  return(log_likehood_poisson(y,x,betas)+log_prior_likelihood(betas))
}

starting_value=rep(0,p)
op=optim(starting_value,log_posterior,gr=NULL,response,covariates,method=c("BFGS"),control=list(fnscale=
posterior_mode=op$par
posterior_cov=-solve(op$hessian)
cat("\n Posterior Mode of betas:",posterior_mode)
cat("\n Posterior Covariance:\n")
kable(posterior_cov)
log_posterior=function(y,x,theta_p,previous_theta,sigma,first=FALSE){
  if(first==TRUE){
    return(log_likehood_poisson(y,x,theta_p))
  }else{
    return(log_likehood_poisson(y,x,theta_p)+dmvnorm(theta_p,mean=previous_theta,sigma=posterior_cov,log
  }
```

```
}
# RWM algorithm
RWMSampler=function(c,y,x,maxit,log_posterior,...){
  theta_series=matrix(nrow = maxit,ncol=p)
  theta_series[1,]=rep(0,p)
  theta_logs=vector(length = maxit)
  theta_logs[1]=log_posterior(y,x,theta_series[1,],first = TRUE)
  for(i in 2:maxit){
    u=runif(1)
    # sample from proposal
    theta_p=rmvnorm(1,mean=theta_series[i-1,],sigma=c*posterior_cov)

    current_log_d=log_posterior(y=y,x=x,theta_p=theta_p,previous_theta=theta_series[i-1,],sigma=c*poste

    alpha=min(1,exp(current_log_d-theta_logs[i-1]))
    if(u<alpha){
      theta_series[i,]=theta_p
      theta_logs[i]=current_log_d

    }else{
      theta_series[i,]=theta_series[i-1,]
      theta_logs[i]=theta_logs[i-1]
    }
  }
  return(list(theta_series=theta_series,theta_logs=theta_logs))

}
result=RWMSampler(1.3,response,covariates,maxit=10000,log_posterior)
theta_samples=result$theta_series
coef_result=colMeans(theta_samples)
colnames=names(data[,-1])
for(i in 1:length(colnames)){
  plot(theta_samples[,i],main=colnames[i],ylab="Value")
}

mu0 = rep(0, times = p)
sigma0 = 100*solve(t(covariates)%*%covariates)

log_post = function(betas, x, y) {

  log_likelihood=rep(0,length(y))
  for(i in 1:length(y)){
    log_likelihood[i]=y[i]*betas%*%x[i,]-exp(betas%*%x[i,])-log(factorial(y[i]))
  }


  logprior = dmvnorm(betas, mean = as.matrix(mu0), sigma0, log=TRUE)
  # since it is log posterior, it is proportional to sum of loglikli&logprior
  # since optim function looks for minimum, minus posterior dist should be used
  return(sum(log_likelihood)+logprior)
}

RMW_sample = function(prev_betas, tuning, sigma, logposterior, ...) {
```

```
  proposal = rmvnorm(1, mean = prev_betas, sigma = tuning*sigma)
  alpha = min(1, logposterior(proposal, ...)/logposterior(prev_betas, ...))
  u = runif(1)
  if (u < alpha) {return(proposal)}
  else {return(prev_betas)}
}

RMW_result = matrix(NA, nrow = 10000, ncol=p)
for (i in 1:10000){
  RMW_result[i,] = RMW_sample(posterior_mode, tuning = 1.3, sigma = posterior_cov, logposterior = log_p
}

par(mfrow = c(1,2))
for (i in 1:p){
  title = paste("Histogram of simulated beta", i)
  xlabel = paste("values of beta", i)
  hist(RMW_result[,i], main=title, xlab = xlabel, breaks=50, freq = F)
  cum_beta = cumsum(RMW_result[,i])
  for (j in 1:10000){
    cum_beta[j] = cum_beta[j] / j
  }
  title2 = paste("Traceplot of beta", i)
  plot(cum_beta, main = title2, xlab = "MCMC Iteration", ylab="Cummulative Estimate")
}
new_bidder=matrix(c(1,1,1,1,0,0,0,1,0.5),nrow = 1)
coef_result=as.matrix(coef_result)
pred=vector(length=10000)
for(i in 1:10000){
  pred[i]=rpois(1,exp(new_bidder%*%coef_result))
}
hist(pred,main = "Histogram of New Bidder Prediction")
prob=length(which(pred==0))/10000
cat("Hte probability of no bidder in this new auction is ",prob)
```