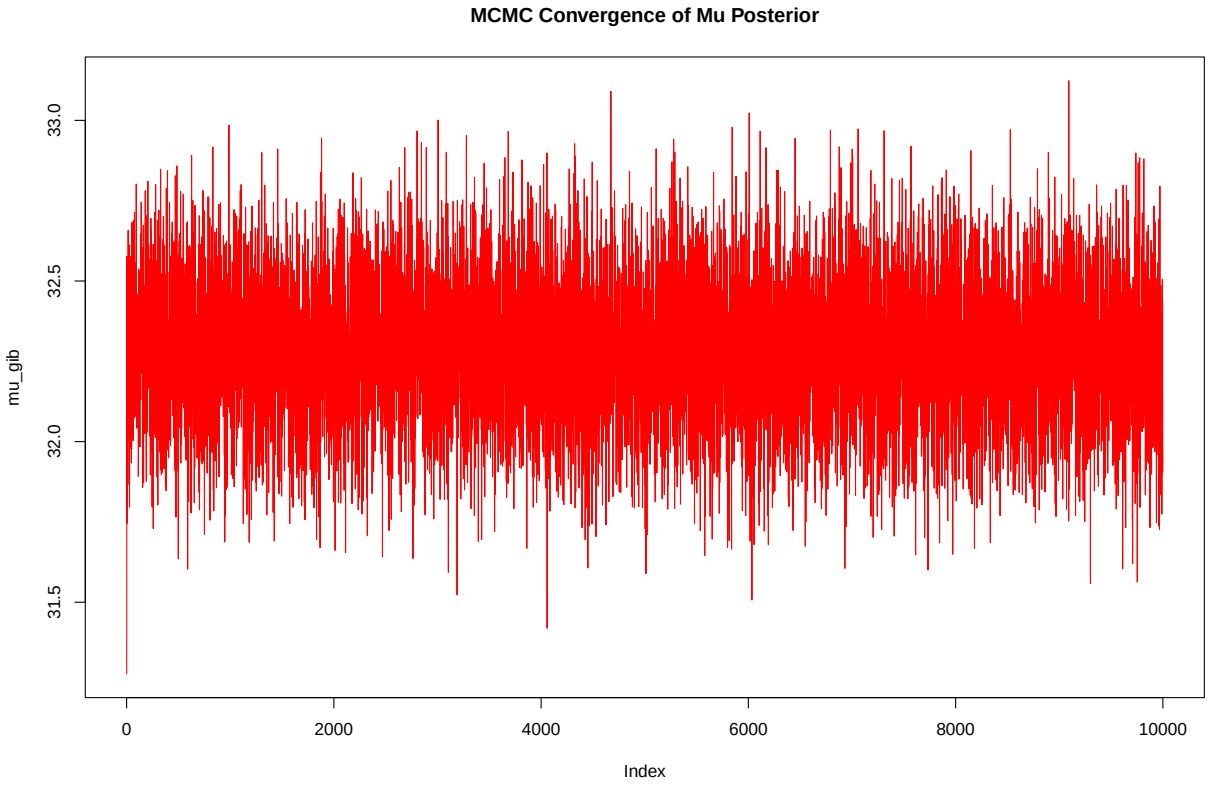
BayesianLab3

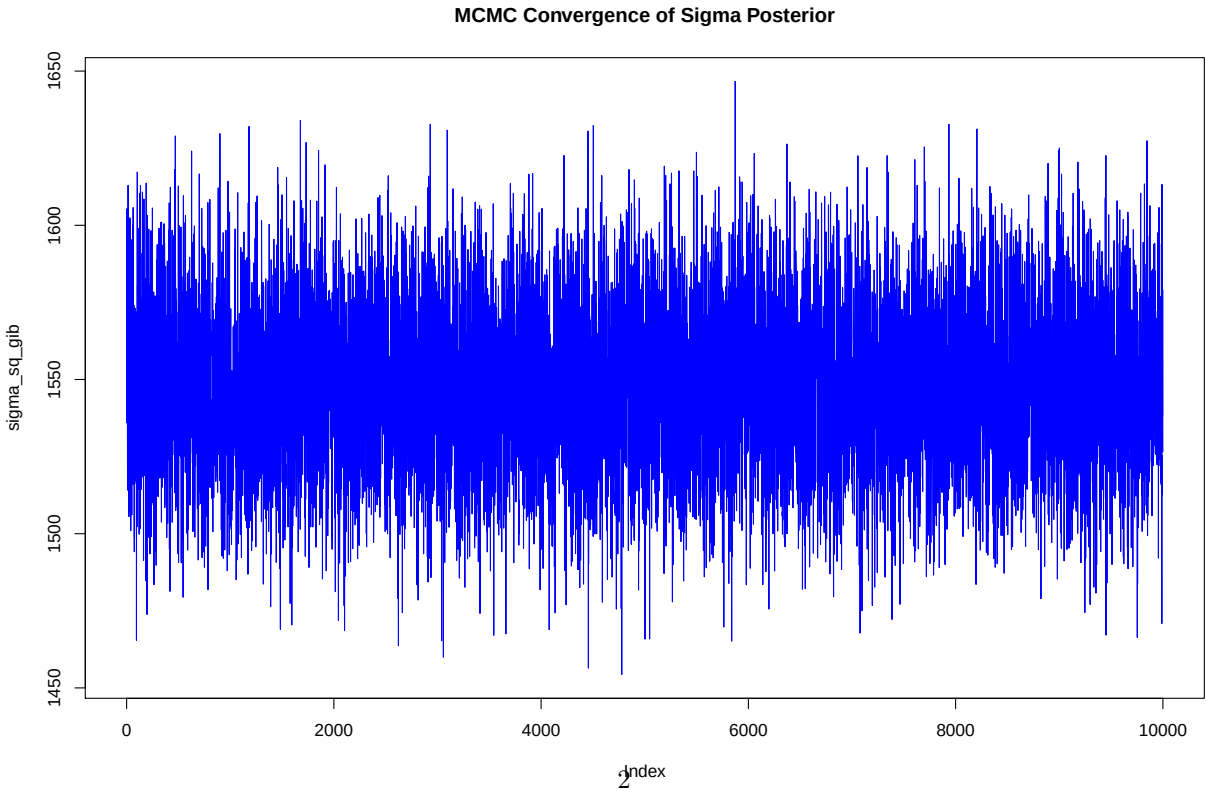
Harshavardhan Subramanian Vinod Kumar Dasari

16/05/2020

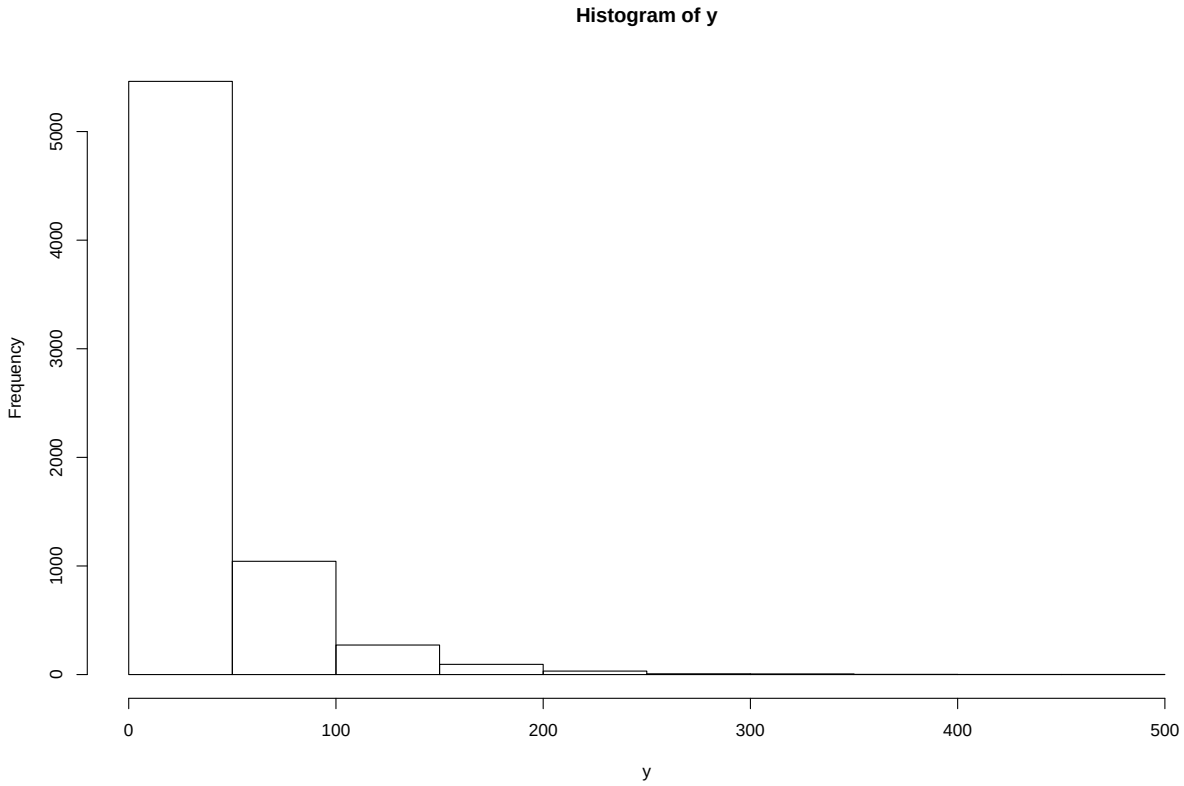
1. Normal model, mixture of normal model with semi-conjugate prior.

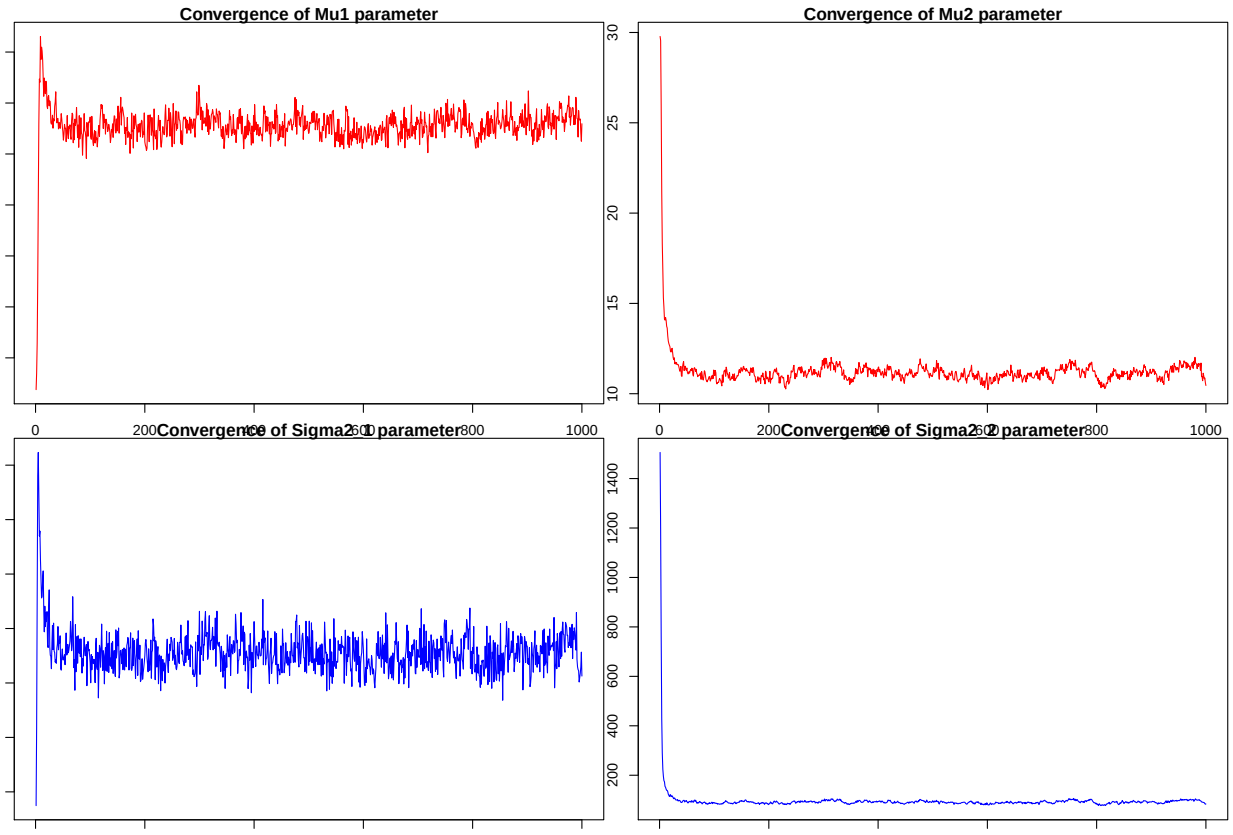
a





b

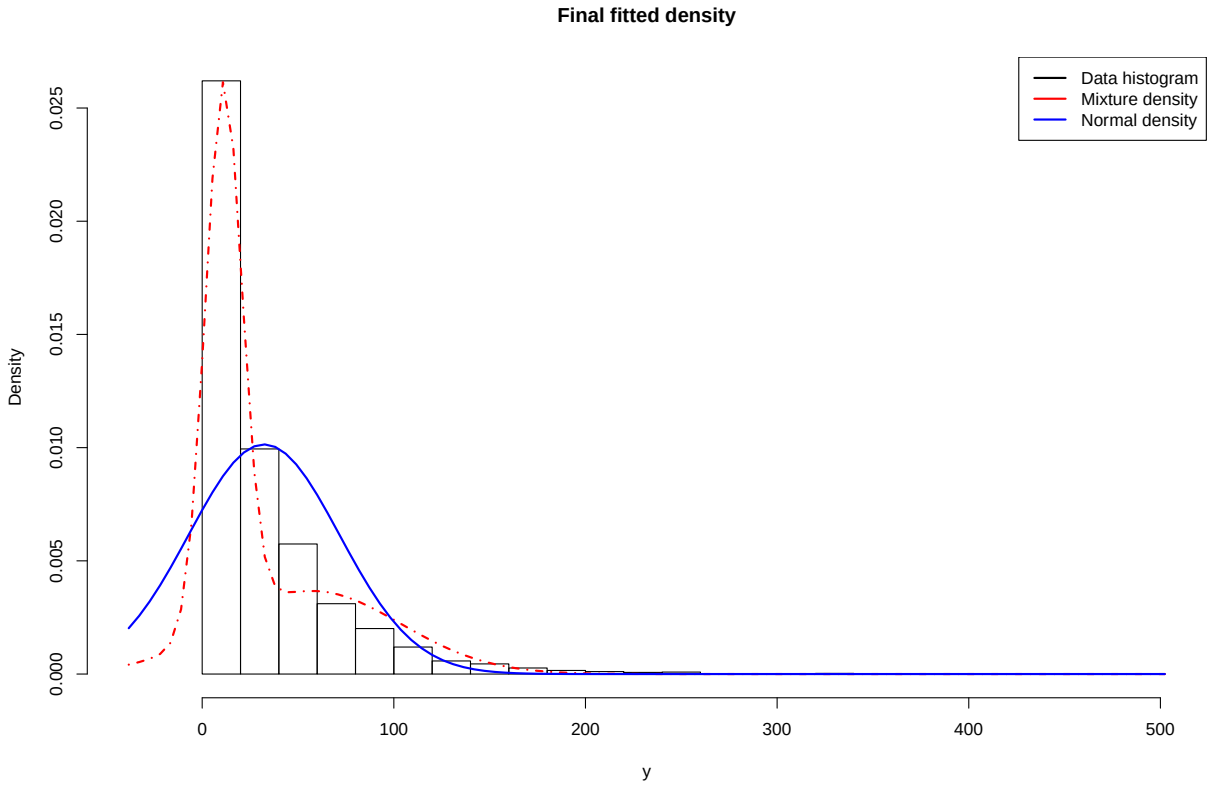

Histogram of y
Frequency
0
1000
2000
3000
4000
5000
0
100
200
300
400
500
y


Convergence of Mu1 parameter
Convergence of Mu2 parameter
Convergence of Sigma2_1 parameter
Convergence of Sigma2_2 parameter
0
200
400
600
800
1000
10
15
20
25
30
200
400
600
800
1000
1200
1400

```
## The gibbs sampler of Mu1 converges to : 57.97246 The gibbs sampler of Mu2 converges to : 10.45539
##
##  The gibbs sampler of Sigma2_1 converges to : 2025.403 The gibbs sampler of Sigma2_2 converges to : 8
## Average of Gibbs Sampler Mu 63.20015  is close to 32.28309  and Average of Gibbs Sampler Sigma2 2067
```

The above obtained values of Mu1,Mu2,Sigma2_1,Sigma2_2 is close to original Mean and Variance of the given data when the average of each parameter is take. Hence the obtained sampler values converges to original Mean and Variance of the Normally distributed data.

c



2. Metropolis Random Walk for Poisson regression.

a

```
##
## Call:
## glm(formula = response ~ covariates, family = "poisson")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5800  -0.7222  -0.0441   0.5269   2.4605
##
## Coefficients: (1 not defined because of singularities)
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            1.07244    0.03077  34.848  < 2e-16 ***
```

```
## covariatesConst             NA         NA      NA       NA
## covariatesPowerSeller -0.02054    0.03678  -0.558   0.5765
## covariatesVerifyID    -0.39452    0.09243  -4.268 1.97e-05 ***
## covariatesSealed       0.44384    0.05056   8.778  < 2e-16 ***
## covariatesMinblem     -0.05220    0.06020  -0.867   0.3859
## covariatesMajBlem     -0.22087    0.09144  -2.416   0.0157 *
## covariatesLargNeg      0.07067    0.05633   1.255   0.2096
## covariatesLogBook     -0.12068    0.02896  -4.166 3.09e-05 ***
## covariatesMinBidShare -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
```

Since the Pvalues for features like VerifyID, Sealed, MajBlem, LogBook, BidShare is less than standard 0.05, these features has significant impact on the prediction.

b

```
## The loglikelihood value for logarithemic posterior is: -4069.919
```

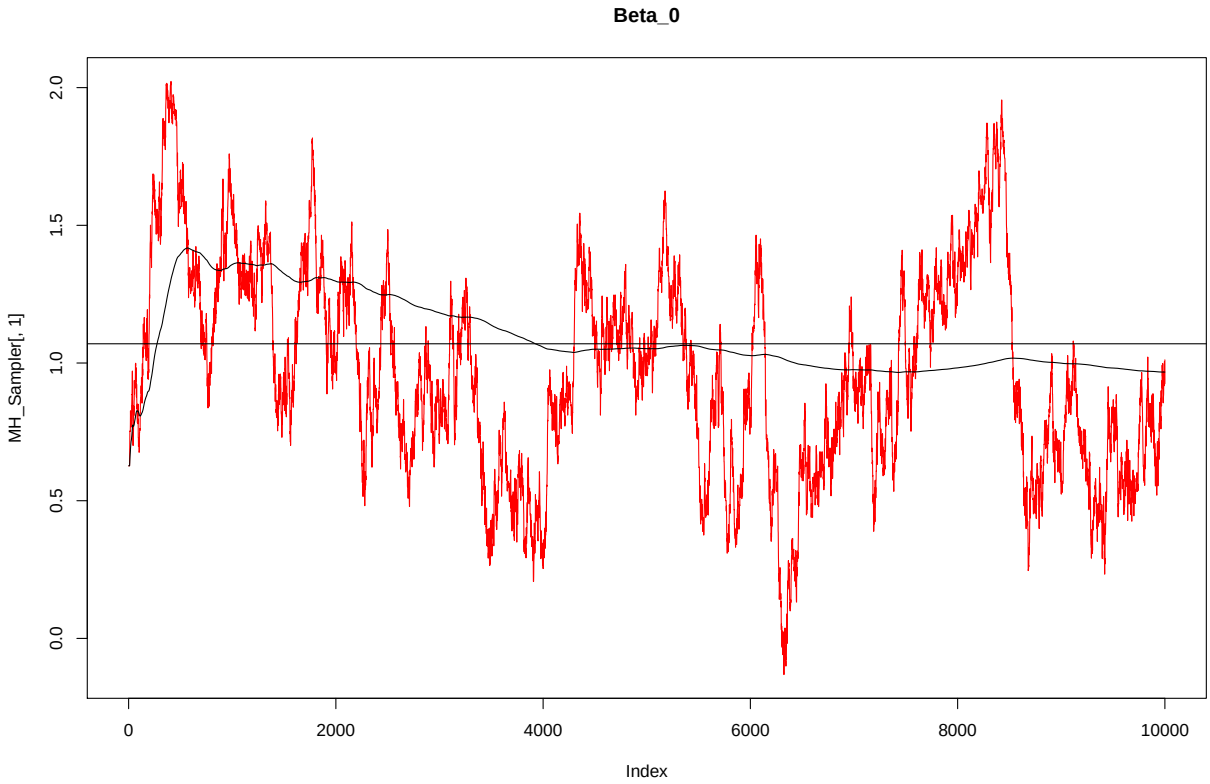
```
## The posterior mode is found by simulation is :
##  1.069841 -0.02051246 -0.393006 0.4435555 -0.05246627 -0.2212384 0.07069683 -0.1202177 -1.891985
```

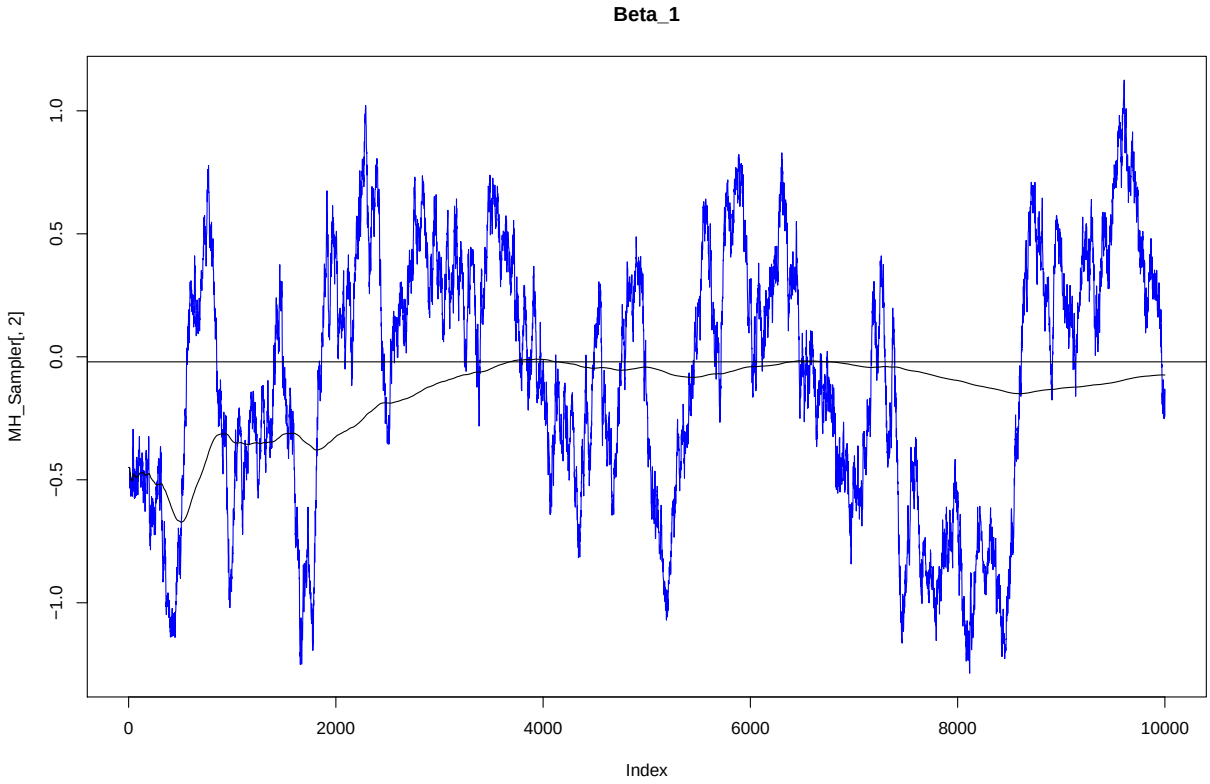
```
## The observed Inverse Hessian matrix evaluated at the posterior mode is :
```

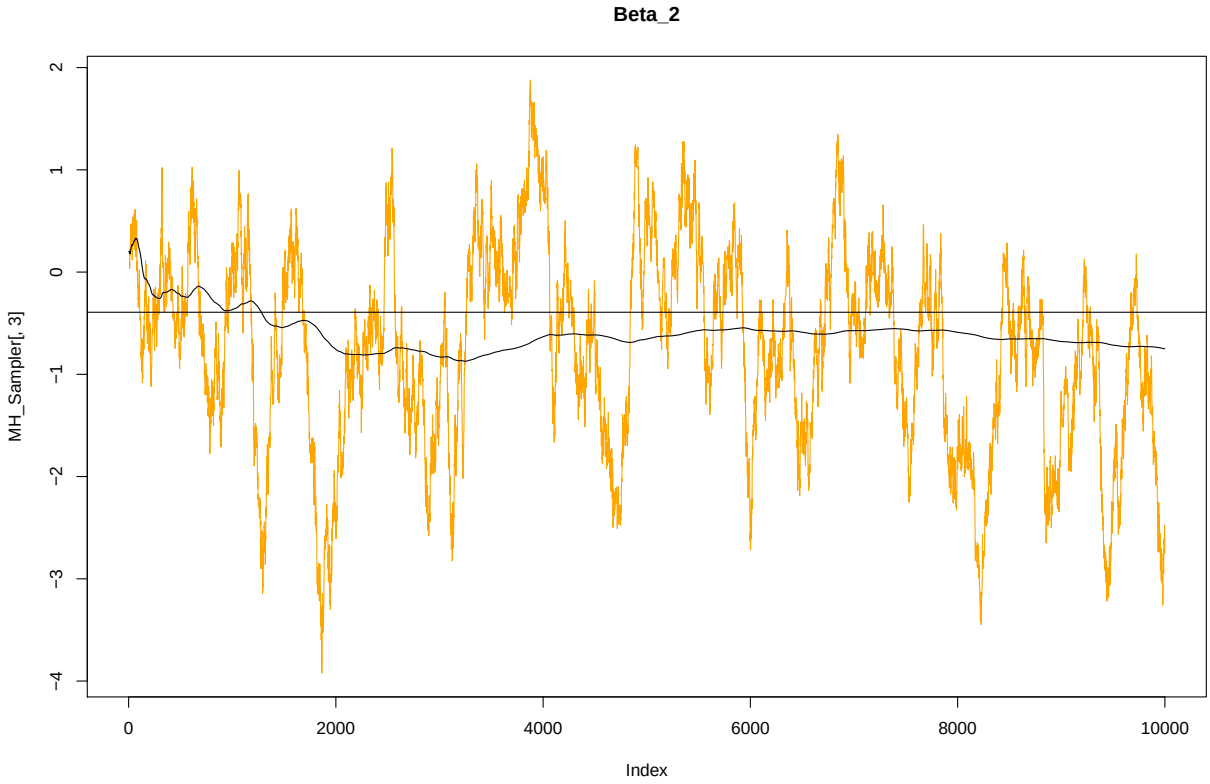
```
##                [,1]          [,2]          [,3]          [,4]
##  [1,]  9.454625e-04 -7.138972e-04 -2.741517e-04 -2.709016e-04
##  [2,] -7.138972e-04  1.353076e-03  4.024623e-05 -2.948968e-04
##  [3,] -2.741517e-04  4.024623e-05  8.515360e-03 -7.824886e-04
##  [4,] -2.709016e-04 -2.948968e-04 -7.824886e-04  2.557778e-03
##  [5,] -4.454554e-04  1.142960e-04 -1.013613e-04  3.577158e-04
##  [6,] -2.772239e-04 -2.082668e-04  2.282539e-04  4.532308e-04
##  [7,] -5.128351e-04  2.801777e-04  3.313568e-04  3.376467e-04
##  [8,]  6.436765e-05  1.181852e-04 -3.191869e-04 -1.311025e-04
##  [9,]  1.109935e-03 -5.685706e-04 -4.292828e-04 -5.759169e-05
##                [,5]          [,6]          [,7]          [,8]
##  [1,] -4.454554e-04 -2.772239e-04 -5.128351e-04  6.436765e-05
##  [2,]  1.142960e-04 -2.082668e-04  2.801777e-04  1.181852e-04
##  [3,] -1.013613e-04  2.282539e-04  3.313568e-04 -3.191869e-04
##  [4,]  3.577158e-04  4.532308e-04  3.376467e-04 -1.311025e-04
##  [5,]  3.624606e-03  3.492353e-04  5.844006e-05  5.854011e-05
##  [6,]  3.492353e-04  8.365059e-03  4.048644e-04 -8.975843e-05
##  [7,]  5.844006e-05  4.048644e-04  3.175060e-03 -2.541751e-04
##  [8,]  5.854011e-05 -8.975843e-05 -2.541751e-04  8.384703e-04
##  [9,] -6.437066e-05  2.622264e-04 -1.063169e-04  1.037428e-03
##                [,9]
##  [1,]  1.109935e-03
##  [2,] -5.685706e-04
##  [3,] -4.292828e-04
```

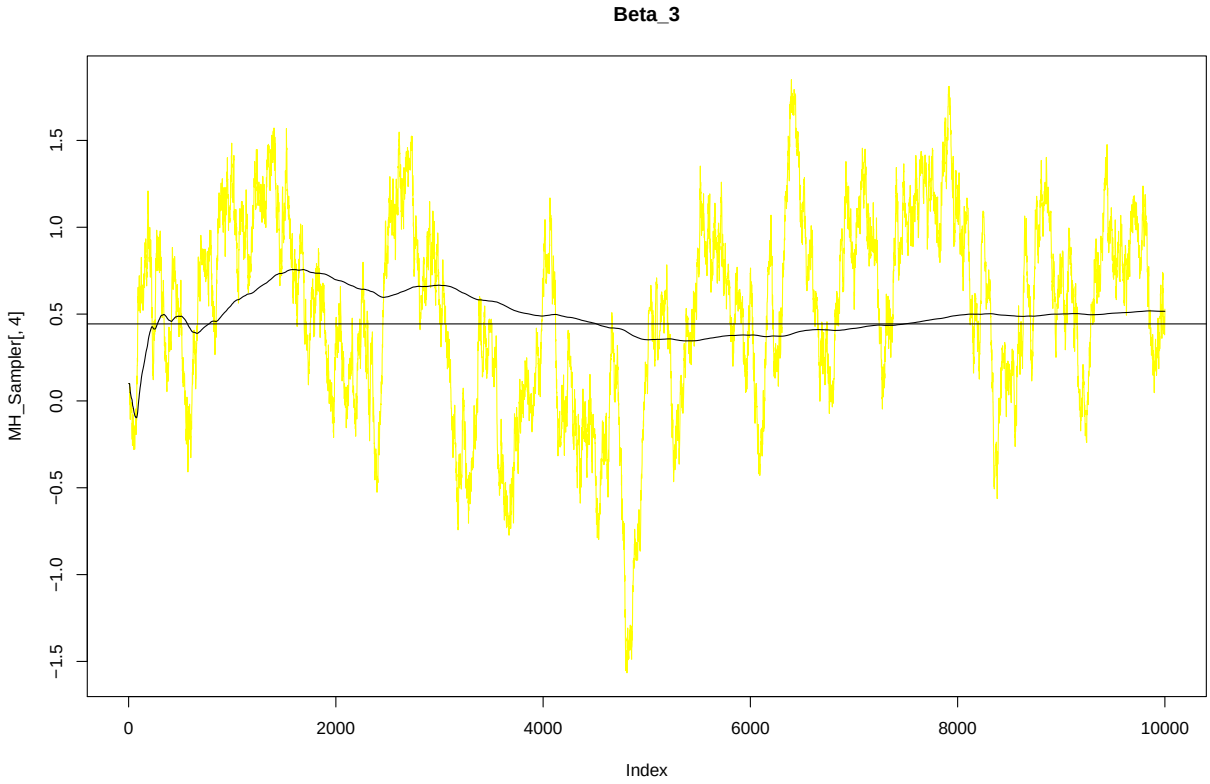
```
##  [4,] -5.759169e-05
##  [5,] -6.437066e-05
##  [6,]  2.622264e-04
##  [7,] -1.063169e-04
##  [8,]  1.037428e-03
##  [9,]  5.054757e-03
```

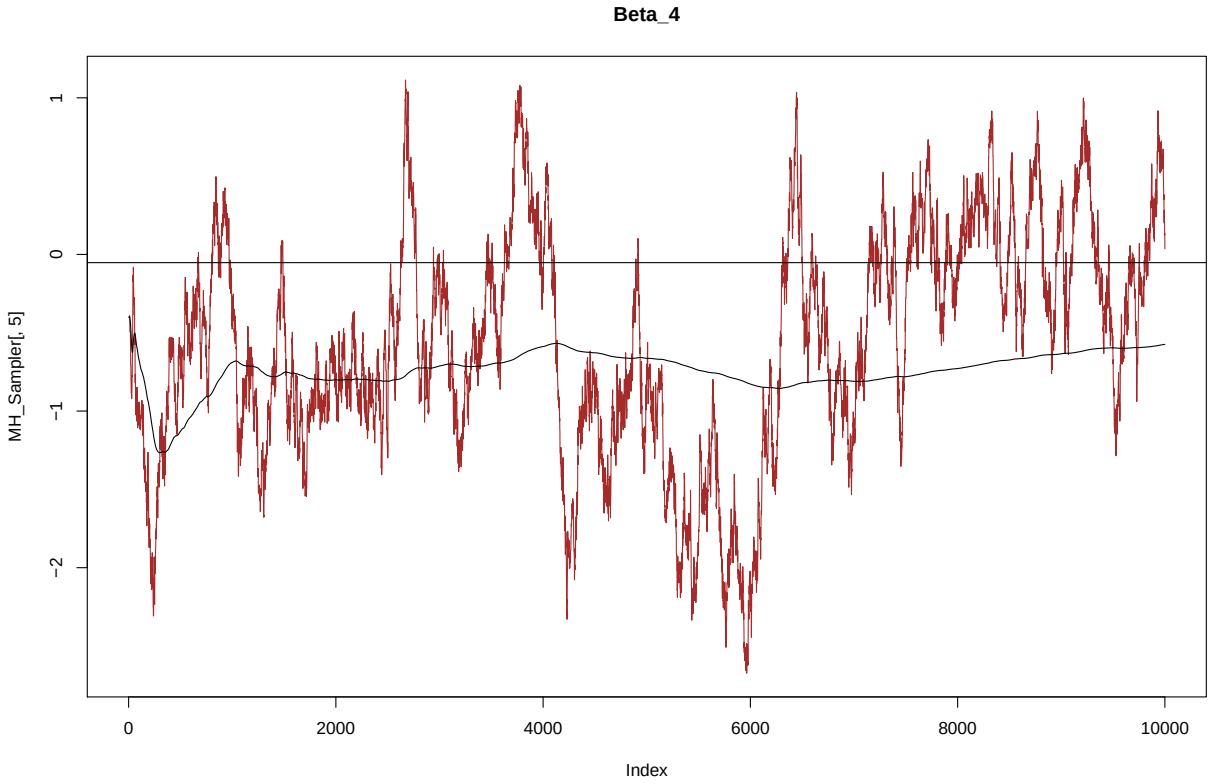
C

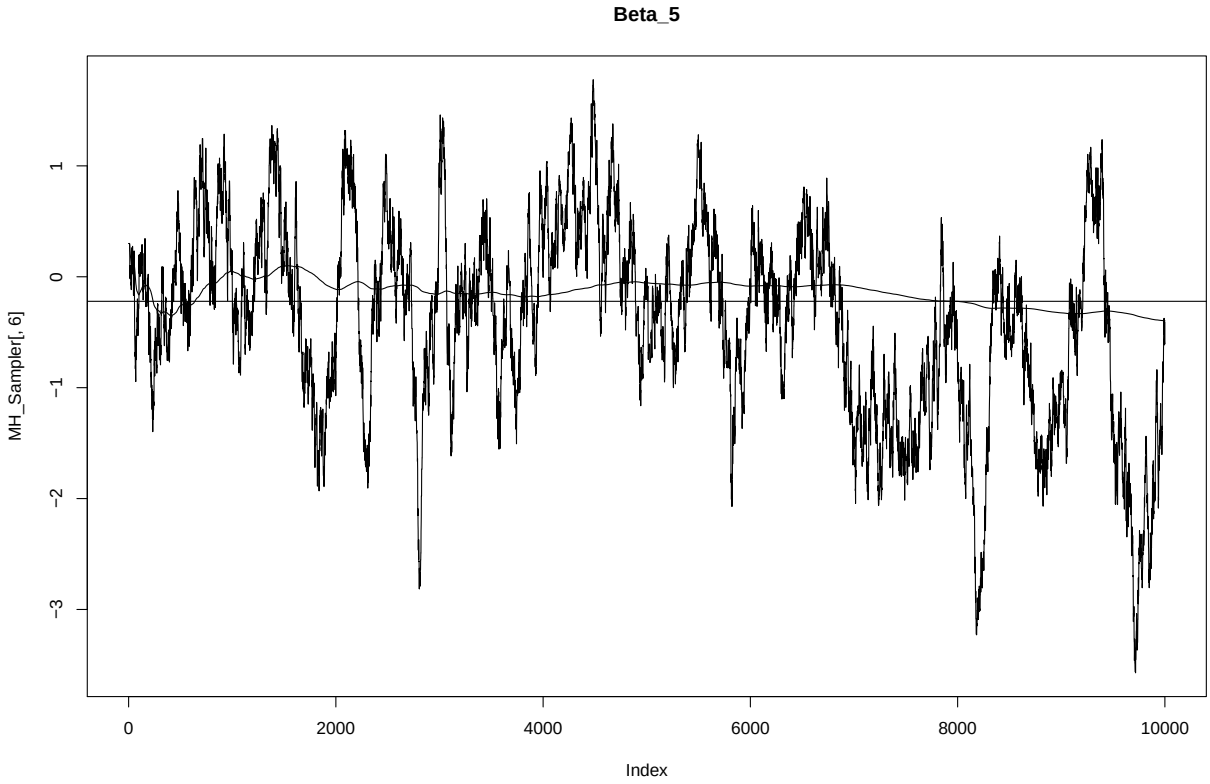

Beta_0
MH_Sampler[, 1]
2.0
1.5
1.0
0.5
0.0
0
2000
4000
6000
8000
10000
Index

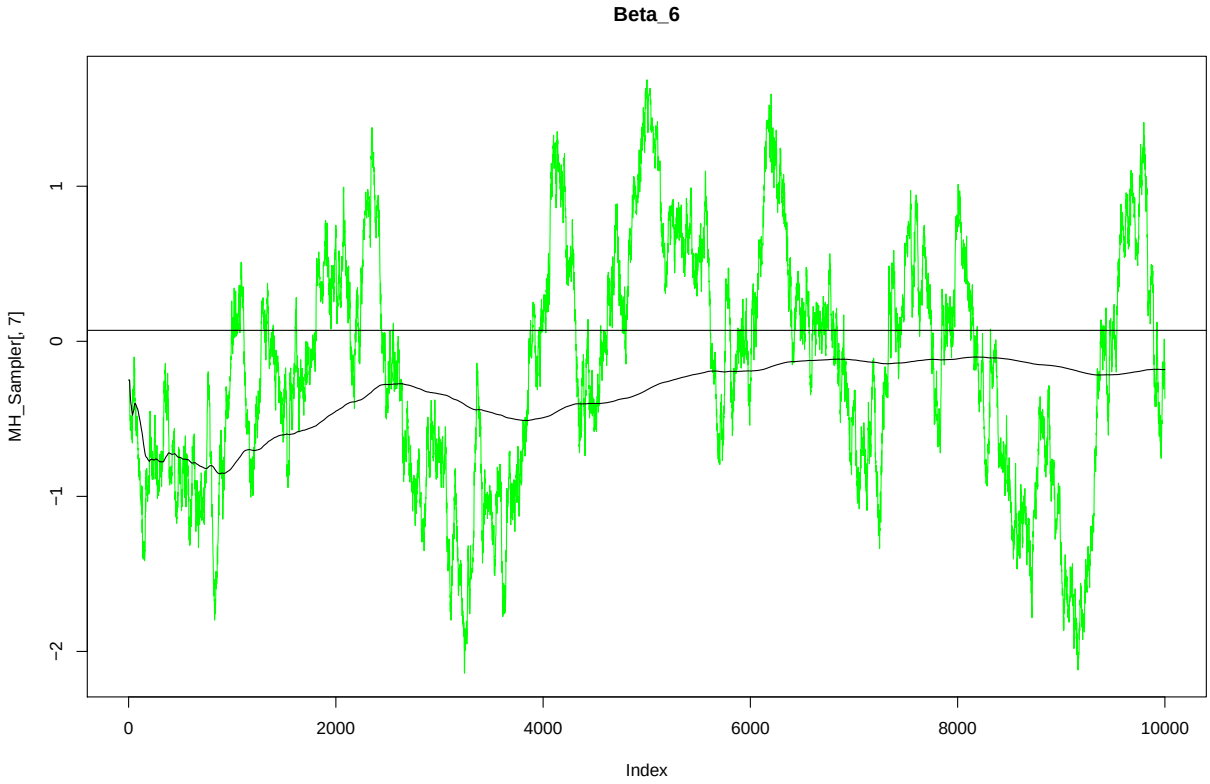

Beta_1
MH_Sampler[, 2]
1.0
0.5
0.0
−0.5
−1.0
0
2000
4000
6000
8000
10000
Index

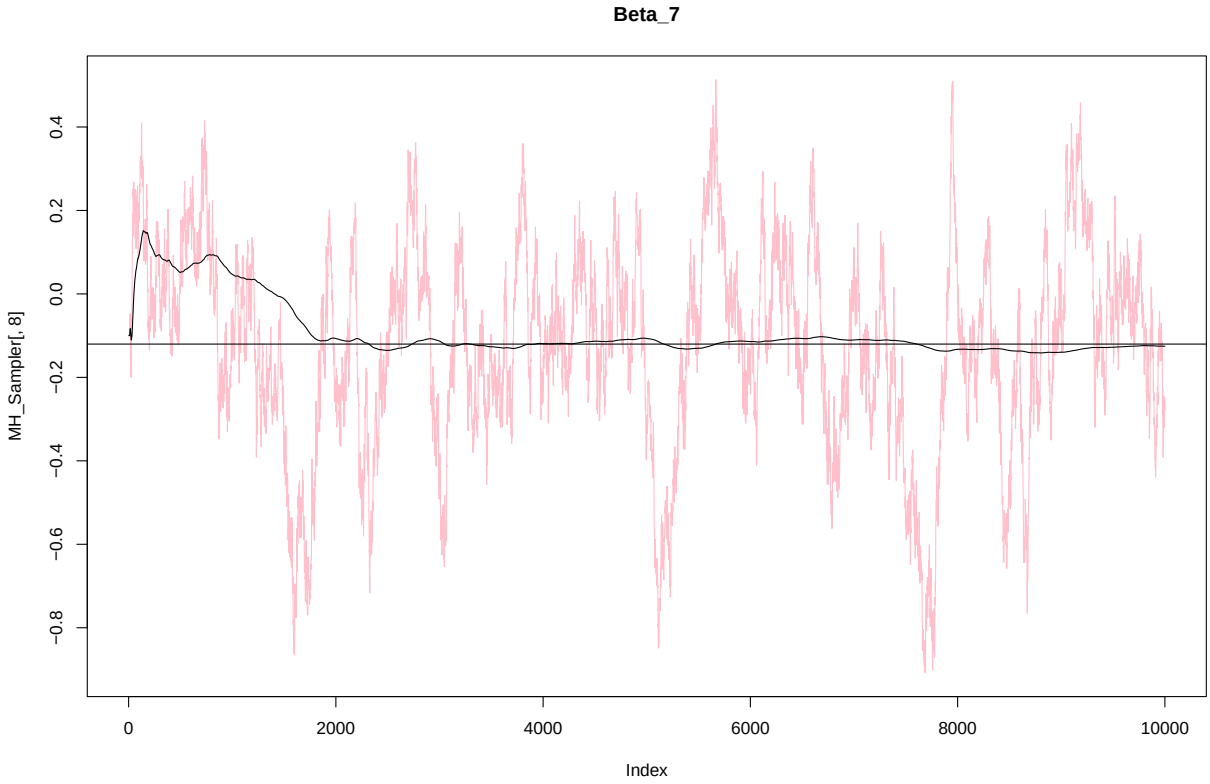

Beta_2
MH_Sampler[, 3]
2
1
0
−1
−2
−3
−4
0
2000
4000
6000
8000
10000
Index

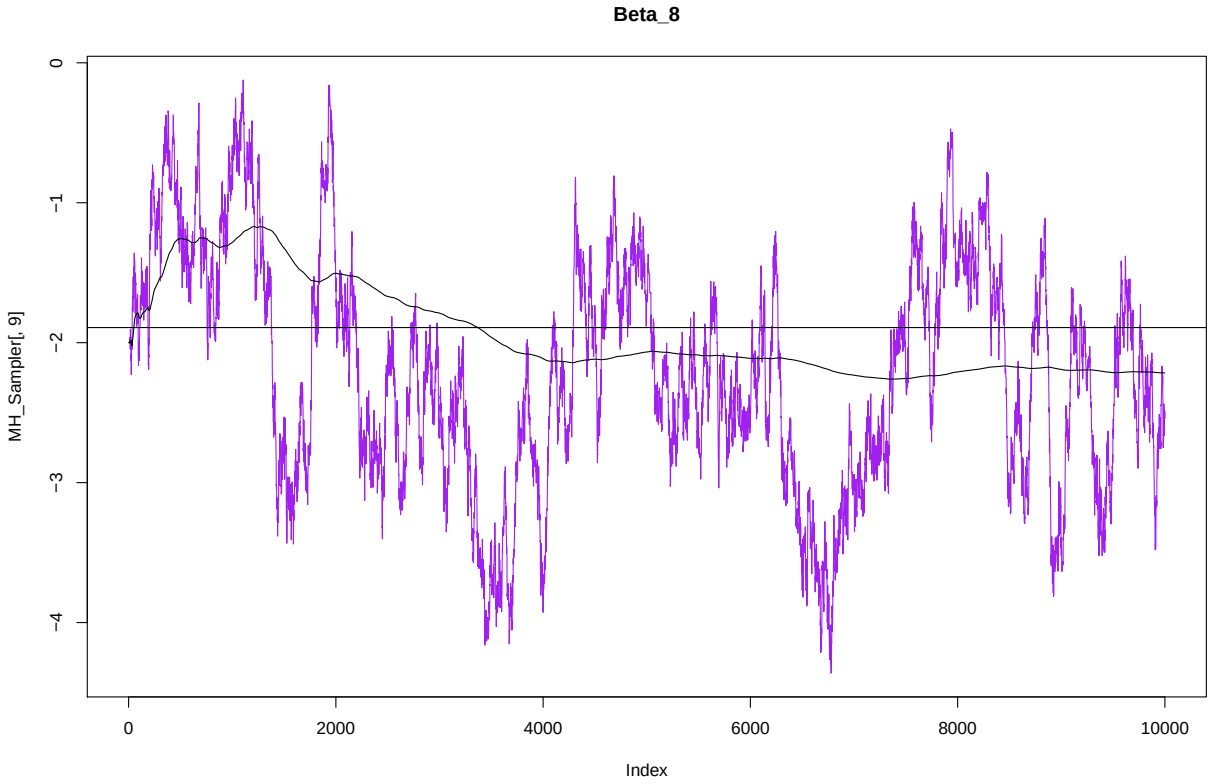

Beta_3
MH_Sampler[, 4]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
2000
4000
6000
8000
10000
Index


Beta_4
MH_Sampler[, 5]
1
0
−1
−2
0
2000
4000
6000
8000
10000
Index


Beta_5
MH_Sampler[, 6]
1
0
−1
−2
−3
0
2000
4000
6000
8000
10000
Index

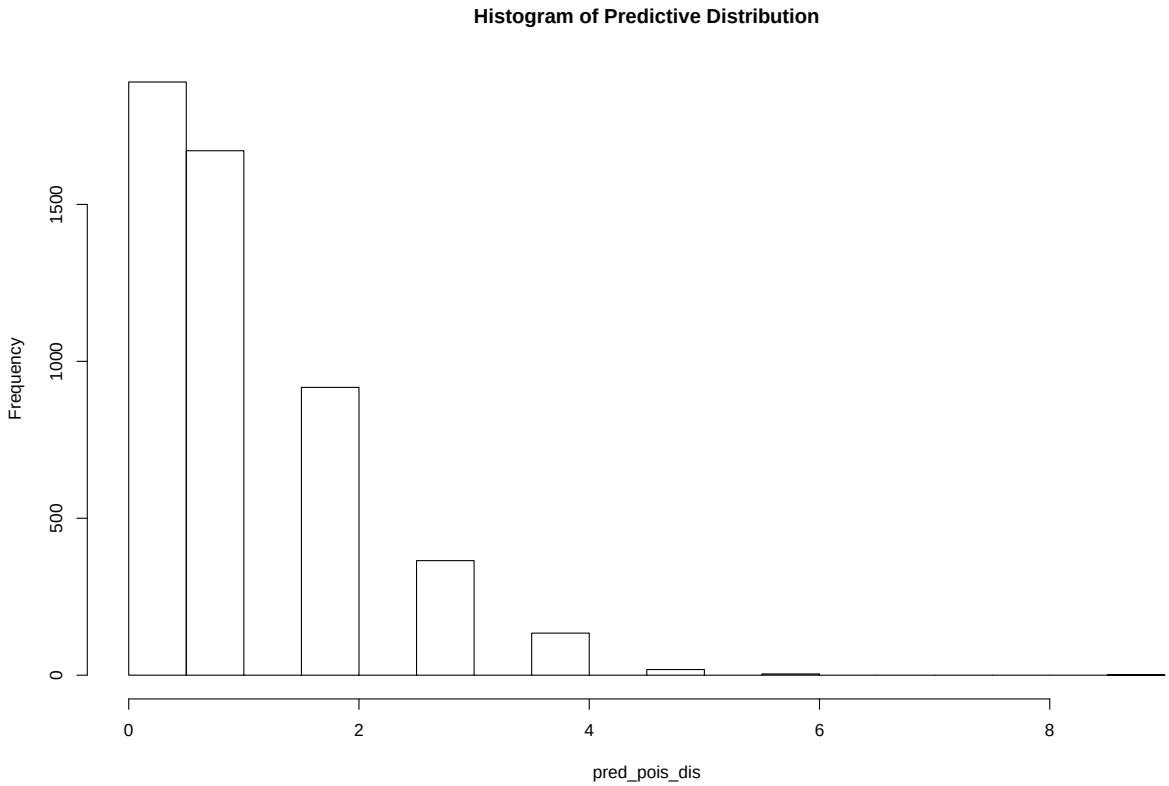

Beta_6
MH_Sampler[, 7]
1
0
−1
−2
0
2000
4000
6000
8000
10000
Index


Beta_7
MH_Sampler[, 8]
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
2000
4000
6000
8000
10000
Index



d

```
## The probability of No bidders in the new auction is found to be: 0.378
```



Appendix

```
knitr::opts_chunk$set(echo = TRUE,warning = FALSE, message = FALSE,fig.width=12, fig.height=8)
rain <- as.data.frame(read.csv("C:\\Users\\shvar\\Desktop\\Bayesian\\rainfall.csv",header = FALSE))
y_bar <- mean(rain[,1])
y <- rain[,1]
n <- length(y)
#sigma_sq <- var(y)
library(geoR)
tau_0 <- 10
mu_0 <- 1.5
v_0 <- 10
v_n <- v_0 + n
sigma_sq_0 <- 2000
mu_gib <- numeric()
sigma_sq_gib <- numeric()
sigma2_temp <- rinvchisq(1,v_0,sigma_sq_0)
for(i in 1:10000)
{
  tau_n <- (tau_0 * sigma2_temp) / (n * tau_0 + sigma2_temp)
  w <- (n / sigma_sq_0) / ((n / sigma_sq_0) + (1 / tau_0 ))
  mu_n <- (w * y_bar) + (1-w) * mu_0
  mu_gib[i] <- rnorm(1,mu_n,tau_n)
  sigma_sq_gib[i] <- rinvchisq(1,v_n,scale = ((v_0 * sigma_sq_0) + sum((rain[,1]-mu_gib[i])^2)) / v_n)
  sigma2_temp <- sigma_sq_gib[i]
  mu_0 <- mu_gib[i]
```

```
}
plot(mu_gib,type = "l",col = "red", main = "MCMC Convergence of Mu Posterior")
plot(sigma_sq_gib,type = "l",col = "blue", main = "MCMC Convergence of Sigma Posterior")

y <- rain[,1]
# Model options
nComp <- 2    # Number of mixture components

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(10,nComp) # Prior mean of mu
tau2Prior <- rep(10,nComp) # Prior std of mu
sigma2_0 <- rep(var(y),nComp) # s20 (best guess of sigma2)
nu0 <- rep(10,nComp) # degrees of freedom for prior on sigma2
# degrees of freedom for prior on sigma2

# MCMC options
nIter <- 1000 # Number of Gibbs sampling draws

# Plotting options
plotFit <- TRUE
lineColors <- c("blue", "green", "magenta", 'yellow')
sleepTime <- 0.1 # 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
nObs <- length(y)
```

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

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

param_conv <- matrix(NA,nrow = nIter, ncol = 6)
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
    param_conv[k,(j+4)] <-  wPrior
    muPost <- wPrior*muPrior + (1-wPrior)*mean(y[alloc == j])
    tau2Post <- 1/precPost
    mu[j] <- rnorm(1, mean = muPost, sd = sqrt(tau2Post))
  }


  # Update sigma2's
  for (j in 1:nComp){
    sigma2[j] <- rScaledInvChi2(1, df = nu0[j] + nAlloc[j], scale = (nu0[j]*sigma2_0[j] + sum((y[alloc =
  }
  param_conv[k,1] <- mu[1]
  param_conv[k,2] <- mu[2]
  param_conv[k,3] <- sigma2[1]
  param_conv[k,4] <- sigma2[2]

  # Update allocation
  for (i in 1:nObs){
    for (j in 1:nComp){
      probObsInComp[j] <- pi[j]*dnorm(y[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

  }

}
par(mar=c(1,1,1,1))
par(mfrow = c(2,2))
plot(param_conv[,1],type = "l",main = "Convergence of Mu1 parameter",col = "red")
plot(param_conv[,2],type = "l",main = "Convergence of Mu2 parameter",col = "red")
plot(param_conv[,3],type = "l",main = "Convergence of Sigma2_1 parameter",col = "blue")
plot(param_conv[,4],type = "l",main = "Convergence of Sigma2_2 parameter",col = "blue")
cat("The gibbs sampler of Mu1 converges to :",mu[1], "The gibbs sampler of Mu2 converges to :",mu[2])
cat("\n The gibbs sampler of Sigma2_1 converges to :",sigma2[1], "The gibbs sampler of Sigma2_2 converg

cat("Average of Gibbs Sampler Mu", mu[1]+mu[2]/2," is close to", mean(y)," and Average of Gibbs Sampler

hist(y, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = "Final fitted density")
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "red")
lines(xGrid, dnorm(xGrid, mean = mean(y), sd = sd(y)), type = "l", lwd = 2, col = "blue")
legend("topright", box.lty = 1, legend = c("Data histogram","Mixture density","Normal density"), col=c(
ebay_data <- read.csv("C:\\Users\\shvar\\Desktop\\Bayesian\\ebaydata.csv")
response <- as.matrix(ebay_data$nBids)
covariates <- as.matrix(ebay_data[,2:ncol(ebay_data)])
glm_mod <- glm(response ~ covariates,family = "poisson")
summary(glm_mod)
library(mvtnorm)
X <- as.matrix(covariates)
Y <- as.vector(response)
mu <- as.vector(rep(0,ncol(X)))
sigma <- 100* solve(t(X) %*% X)
#diag(sigma) <- tau2
# prior
beta_vect <- rnorm(ncol(X),mu,diag(sigma))
log_post <- function(beta_vect,X,Y,sigma)
{
  loglikelihood <- sum(Y * (X %*% beta_vect) - exp(X %*% beta_vect))
  logprior <- dmvnorm(beta_vect,mean = rep(0,length(beta_vect)),sigma = sigma, log=TRUE)
  logpost <- loglikelihood + logprior
  return(logpost)
}
loglikelihood_val <- log_post(beta_vect,X,Y,sigma)
```

```r
cat("The loglikelihood value for logarithemic posterior is:",loglikelihood_val)

pars <- as.vector(rep(0,ncol(X)))
post_mod <- optim(pars,log_post,gr = NULL,X,Y,sigma,method = "BFGS",control=list(fnscale=-1),hessian=TRU
beta_tilda <- post_mod$par
cat("The posterior mode is found by simulation is :\n",beta_tilda)
InvHess <- -solve(post_mod$hessian)
cat("The observed Inverse Hessian matrix evaluated at the posterior mode is :\n")
InvHess
#
# beta_post <- rmvnorm(1000,mean = beta_tilda, sigma = InvHess)
# plot(density(beta_post))
RWMSampler <- function(Myfunction,C, ...)
{
  #beta_draw <- Myfunction(beta_vect, ...)
  theta_prev <- c(0.6267,-0.45,0.20,0.1,-0.4,0.3,-0.25,-0.1,-2)
  theta_mat <- matrix(NA,10000,ncol(X))
  theta_mat[1,] <- theta_prev
  c1 <- 0
  c2 <- 0
  for(i in 2:10000)
  {
    #i <- 2
  theta_curr <- theta_mat[i-1,]
  theta_p <- as.vector(rmvnorm(1,mean = theta_curr, sigma = C * InvHess))
  # log_post(theta_p,X,Y,sigma)
  num <- Myfunction(theta_p, ...)
  denom <- Myfunction(theta_prev,...)
  ratio <- exp(num - denom)
  alpha <-  min(1,ratio)
  U <- runif(1,0,1)
  if(U < alpha)
  {
    theta_mat[i,] <- theta_p
    c1 <- c1 + 1
  }
  else
  {
    theta_mat[i,] <- theta_mat[i-1,]
    c2 <- c2 + 1
  }
  }
  return(theta_mat)

}
MH_Sampler <- as.data.frame(RWMSampler(log_post,C=1.3,X,Y,sigma))
par(mar=c(5.1, 4.1, 4.1, 2.1))
par(mfrow=c(1,1))
plot(MH_Sampler[,1],col = "red",type = "l",main = "Beta_0")
points((cumsum(MH_Sampler[,1])/seq(1,10000,1)),type = "l")
abline(h = beta_tilda[1])
plot(MH_Sampler[,2],col = "blue",type = "l",main = "Beta_1")
points((cumsum(MH_Sampler[,2])/seq(1,10000,1)),type = "l")
```

```
abline(h = beta_tilda[2])
plot(MH_Sampler[,3],col = "orange",type = "l",main = "Beta_2")
points((cumsum(MH_Sampler[,3])/seq(1,10000,1)),type = "l")
abline(h = beta_tilda[3])
plot(MH_Sampler[,4],col = "yellow",type = "l",main = "Beta_3")
points((cumsum(MH_Sampler[,4])/seq(1,10000,1)),type = "l")
abline(h = beta_tilda[4])
plot(MH_Sampler[,5],col = "brown",type = "l",main = "Beta_4")
points((cumsum(MH_Sampler[,5])/seq(1,10000,1)),type = "l")
abline(h = beta_tilda[5])
plot(MH_Sampler[,6],col = "black",type = "l",main = "Beta_5")
points((cumsum(MH_Sampler[,6])/seq(1,10000,1)),type = "l")
abline(h = beta_tilda[6])
plot(MH_Sampler[,7],col = "green",type = "l",main = "Beta_6")
points((cumsum(MH_Sampler[,7])/seq(1,10000,1)),type = "l")
abline(h = beta_tilda[7])
plot(MH_Sampler[,8],col = "pink",type = "l",main = "Beta_7")
points((cumsum(MH_Sampler[,8])/seq(1,10000,1)),type = "l")
abline(h = beta_tilda[8])
plot(MH_Sampler[,9],col = "purple",type = "l",main = "Beta_8")
points((cumsum(MH_Sampler[,9])/seq(1,10000,1)),type = "l")
abline(h = beta_tilda[9])
X_d <- t(c(1,1,1,1,0,0,0,1,0.5))
pred_pois_dis <- numeric()
for(i in 1:5000)
{
pred_pois_dis[i] <- rpois(1,lambda = t(MH_Sampler[i,]) %*% X_d)
}
prob_no_bid <- length(which(pred_pois_dis == 0))/5000
cat("The probability of No bidders in the new auction is found to be:", prob_no_bid)
hist(pred_pois_dis, main= "Histogram of Predictive Distribution")
```