Bayesian Learning Lab 03

Rakhshanda Jabeen

10/05/2020

Contents

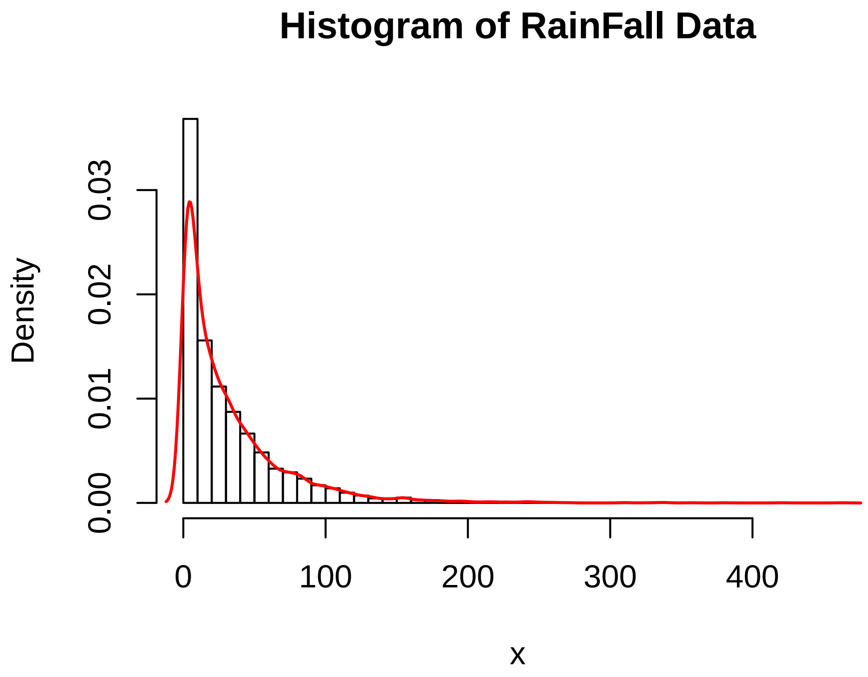
1. Normal Model and Mixture of Normal Model with Semi-conjugate Prior **2**
1.1. Normal Model 2
1.1.1. Gibbs Sampler to Simmulate from Joint Posterior 2
1.1.2. Convergence of Gibbs Sampler 5
1.2. Mixture Normal Model 7
1.2.1. Convergence of the Sampler 8
1.3. Graphical Comparison 9

2. Metropolis Random Walk for Poisson regression. **10**
2.1. Maximum Likelihood Estimator of β 10
2.2. Bayesian Analysis of the Poisson Regression 11
2.3. Posterior of β using MetroPolis Hasting Sampling 12
2.4. Predictive Distribution 16

Appendix **17**

1. Normal Model and Mixture of Normal Model with Semi-conjugate Prior

The data rainfall 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). Our task is to analyze the data using the following normal models and mixture of normals using Gibbs sampling.



1.1. Normal Model

Assuming that the daily precipitations $y_1, y_2, ..., y_n \sim \mathcal{N}\big(\mu, \sigma^2\big)$ are identically independent and normally distributed where both μ and σ^2 are unknown and defined as follows:

$$\mu \sim \mathcal{N}(\mu_0, \tau_0^2)$$
$$\sigma^2 \sim Inv - \chi^2(v_0, \sigma_0^2)$$

1.1.1. Gibbs Sampler to Simmulate from Joint Posterior

In this task we have to implement a Gibbs sampler that simulates from the joint posterior $p(\mu, \sigma^2 | y_1, y_2, ..., y_n)$ whilst the full conditional posteriors are as follows:

$$\mu | \sigma^2, X \sim \mathcal{N}(\mu_n, \tau_n^2)$$

where,

$$\tau_n^2 = \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\tau_0^2}}$$

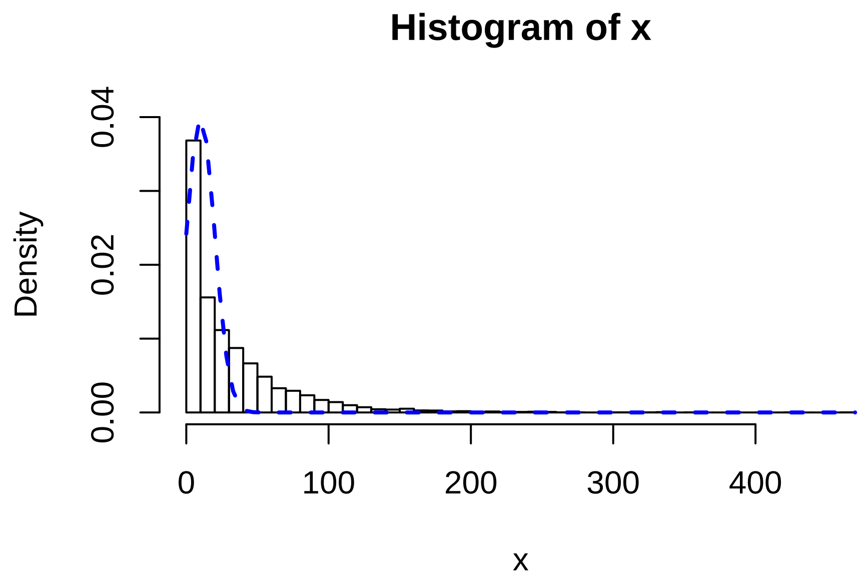
$$\mu_n = w\bar{x} + (1-w)\mu_0$$

$$w = \frac{\frac{n}{\sigma^2}}{\frac{n}{\sigma^2} + \frac{1}{\tau_0^2}}$$

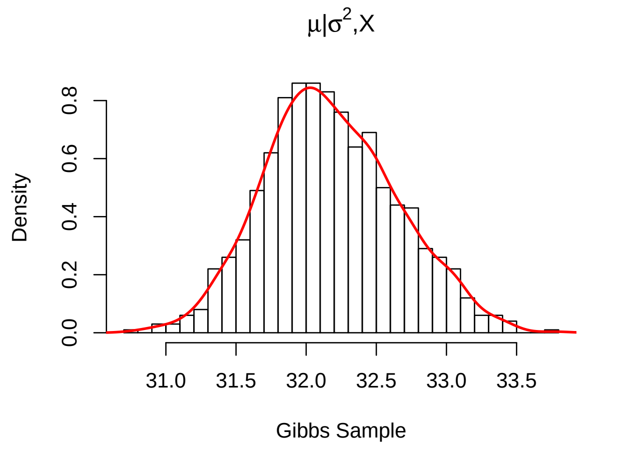
$$\sigma^2|\mu, X \sim Inv - \chi^2\left(v_n, \frac{v_0\sigma_0^2 + \sum_{i=1}^{n}(X_i-\mu)^2}{n+v_0}\right)$$

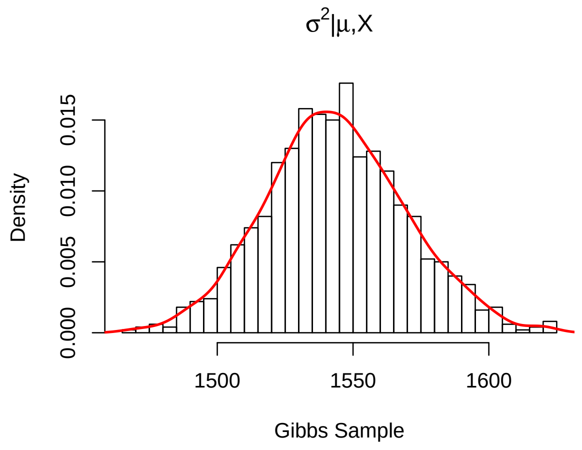
We have selected the following values of prior hyperparameters as we can see in the plot below that this choice of prior is somehow showing the same trend as original observations in rainfall data.

- $\mu_0 = 15$
- $\tau_0^2 = 30$
- $v_0 = 10$
- $\sigma_0^2 = 70$

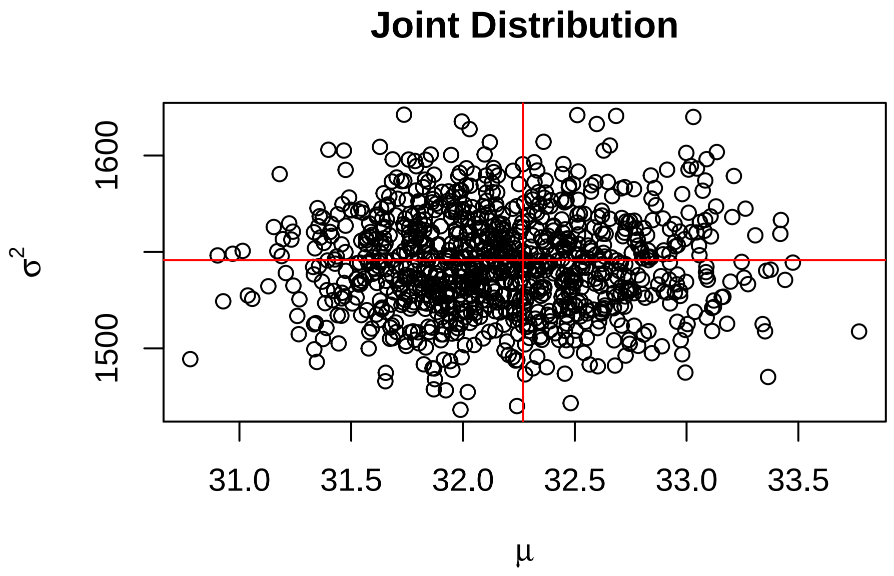


We can visualize the conditional distributions of μ and σ^2 in the following histograms.

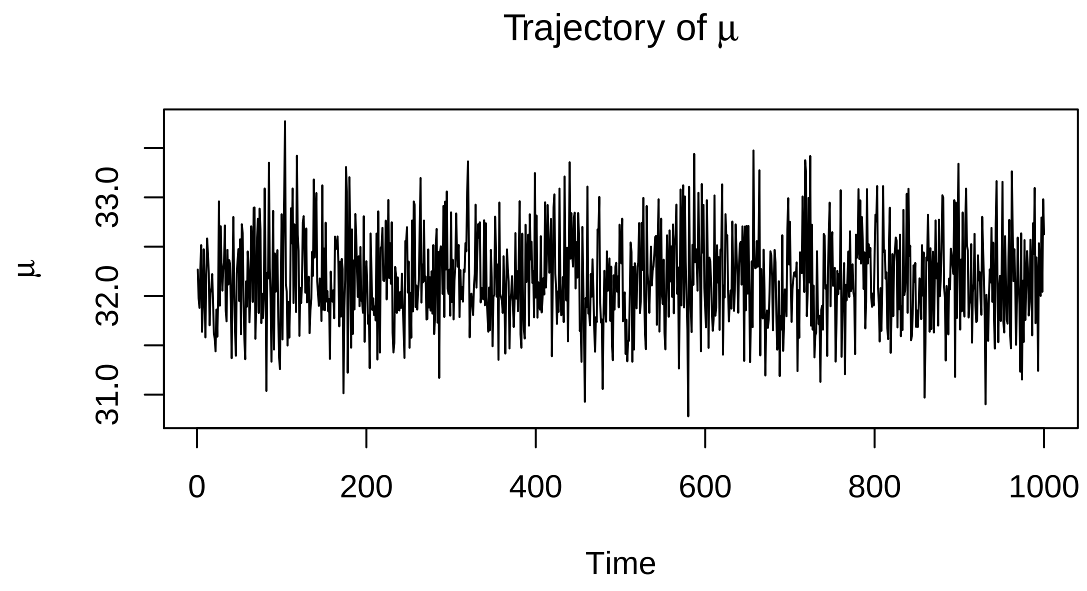


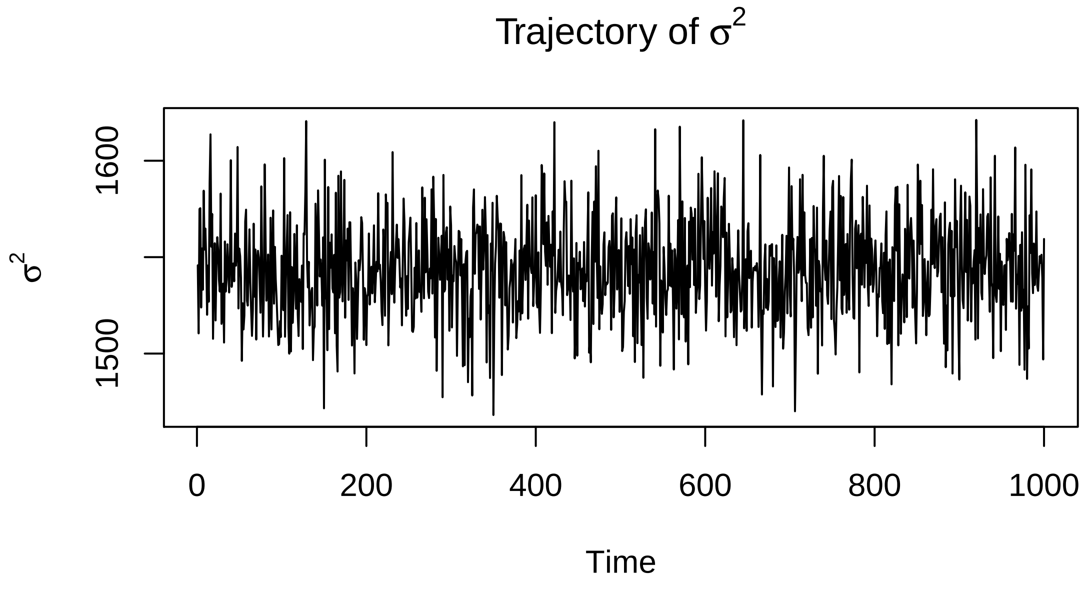


The following plot represents the joint distributions of of the simmulations drawn from the Gibbs sampler. The red lines represents the actual mean and variance of the data. It is evident from the plot that drawn simmulations are centered at these red lines intersection point.



1.1.2. Convergence of Gibbs Sampler

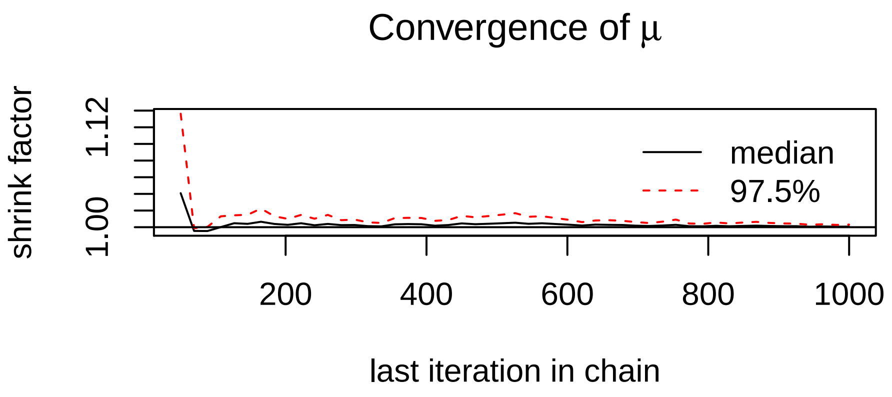




In the above shown trajectories of the parameters we can see that markov chain is traversing in a similar fashion thus we can infer that it is showing convergence to a stationary distribution.
As we can see in both the time series plots that there is no quick descend or ascend of values from the starting point so we can say that there is no noticeable burn-in period. Still we should discard first half of the initial values to deminish the influence of the initial values.

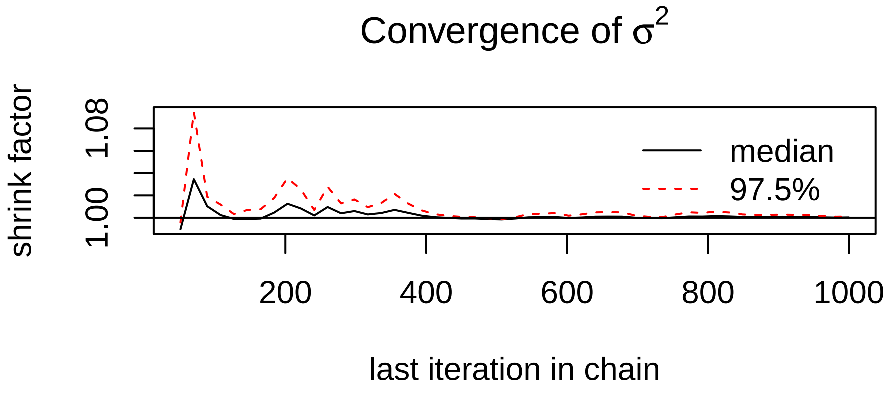
Analysis of Convergence With Gelman-Rubin Diagnostic

Gelman-Rubin diagnosticto is an approach to monitor convergence of MCMC output in which two or more parallel chains are run with overdispersed initial values relative to the posterior distribution. Convergence is diagnosed when the chains forget their initial values, and the output from all chains is identicle because if all the chains converged to the stationary distribution then the variabiity between the curves should be very small. In order to check cinvergence of chains with Gelman-Rubin Diagnositics The potential scale factor should be a small number closer to 1.



```
Potential scale reduction factors:

     Point est. Upper C.I.
[1,]          1          1
```



```
Potential scale reduction factors:

     Point est. Upper C.I.
[1,]          1          1
```

In the above plots we can see that after initial 100-150 values the shrinkage factor has reduced to 1 moreover the potential scale factor is also 1 for both the parameters so we have probably reached to the convergence.

1.2. Mixture Normal Model

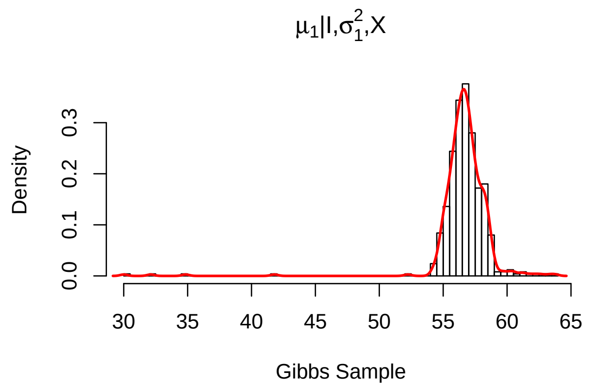
In this task we assume that the the daily precipitation data follows a two component mixture of normal models:

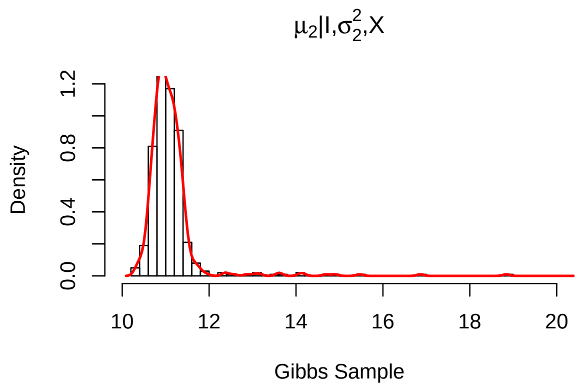
$$p(y_i|\mu, \sigma^2, \pi) = \pi\mathcal{N}(y_i|\mu_1, \sigma_1^2) + (1-\pi)\mathcal{N}(y_i|\mu_2, \sigma_2^2)$$

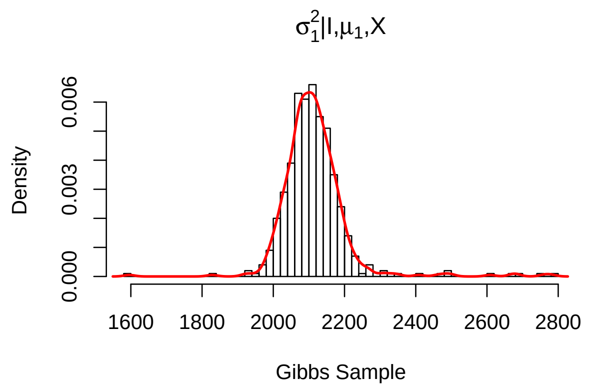
Where, $\mu = (\mu_1, \mu_2)$ and $\sigma^2 = (\sigma_1^2, \sigma_1^2)$. We are using the Gibbs ampling to analyze the daily precipitation data. We set the following values for the prior hyperparameters:

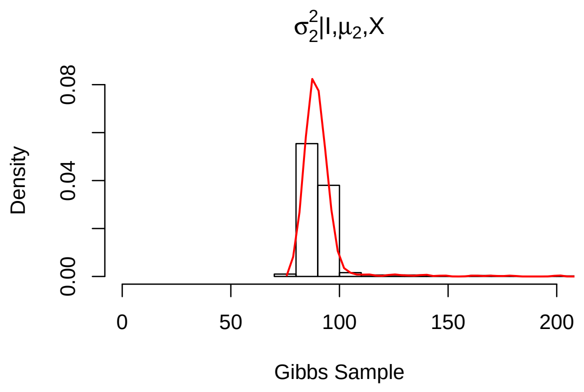
- $\alpha = 15$
- $\mu_0 = c(0, 0)$
- $\tau_0^2 = (10, 10)$
- $v_0 = 10$
- $\sigma_0^2 =$ variance of actual data

Conditional Distribution of Parameter

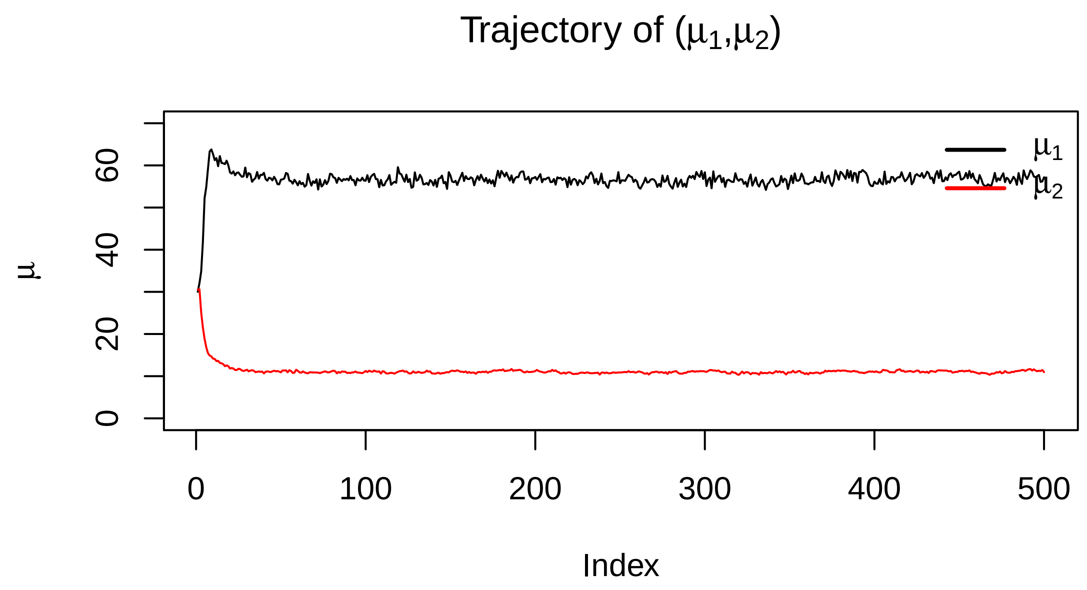


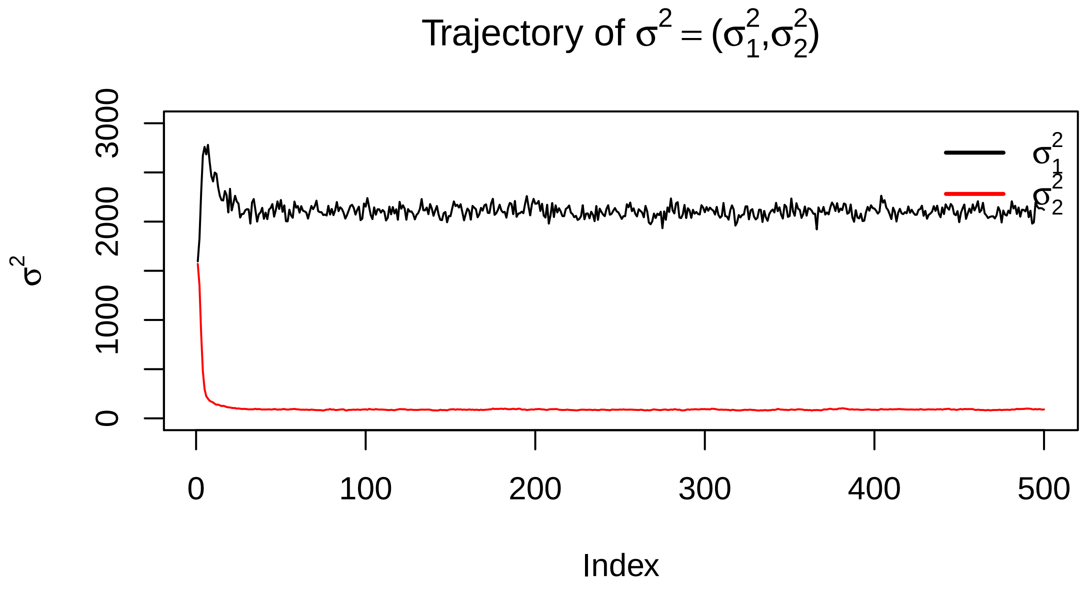






1.2.1. Convergence of the Sampler

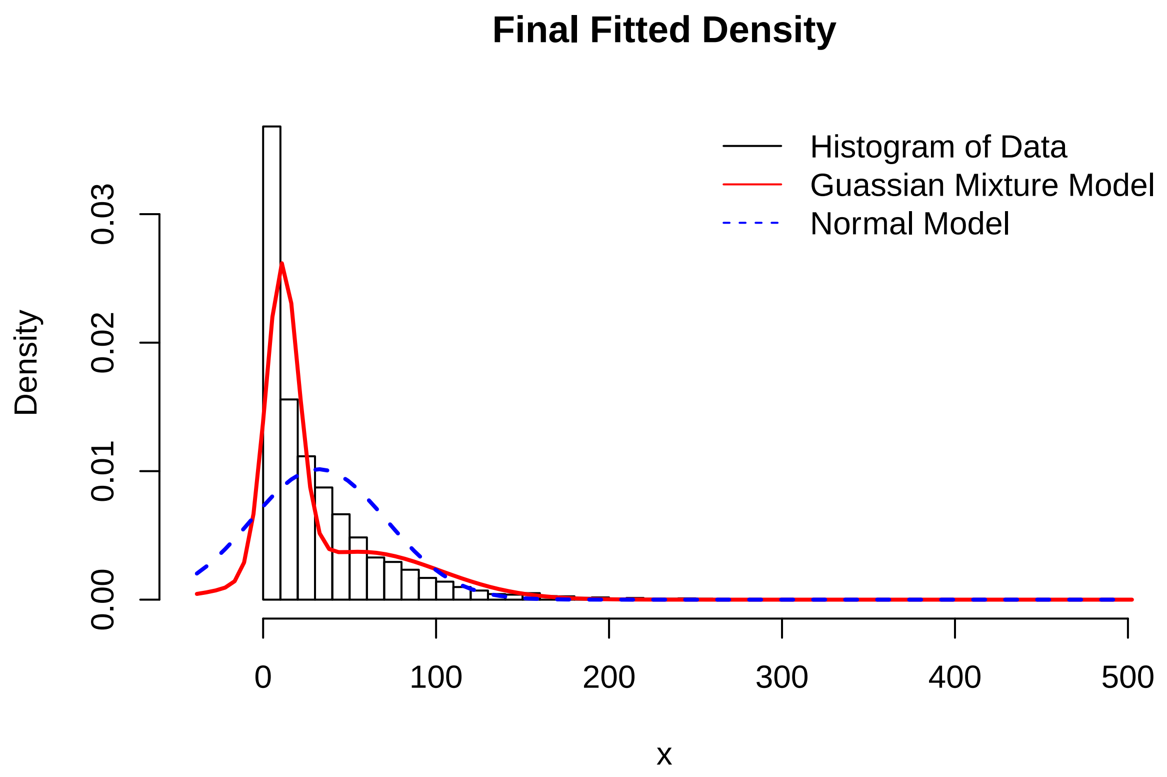




In the above shown trajectories of the parameter μ_1 and σ_1^2 that there is a quick ascend of values from the starting point. Whilst in the trajectories of μ_2 and σ_2^2 there is a quick descenf of values from the starting point. Thus we can see a notice able burn-in period of initial 10-20 samples.

we can see that markov chain is traversing in a similar fashion after the burn-in period thus we can infer that it is showing convergence to a stationary distribution.

1.3. Graphical Comparison



We can clearly see in the visual inspection of both fitted densities, normal model and two component mixture of normal model that mixture of normal model is predicting the actual trend of data quite efficiently. This is because our data is not uni modal.

2. Metropolis Random Walk for Poisson regression.

In this task our dataset contains observations from 1000 eBay auctions f coins. The response variable is nBids and records the number of bids in each auction. We are considering the following poisson regression model:

$$y_i|\beta \sim \text{Poisson}[exp(X_i^T\beta)], \quad i = 1, 2, ..., n$$

2.1. Maximum Likelihood Estimator of β

```
model = glm(formula = nBids ~0+.,data = data,family = "poisson")
summary(model)
```

```
Call:
glm(formula = nBids ~ 0 + ., family = "poisson", data = data)

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|)
Const        1.07244    0.03077  34.848 < 0.0000000000000002 ***
PowerSeller -0.02054    0.03678  -0.558               0.5765
VerifyID    -0.39452    0.09243  -4.268            0.0000197 ***
Sealed       0.44384    0.05056   8.778 < 0.0000000000000002 ***
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            0.0000309 ***
MinBidShare -1.89410    0.07124 -26.588 < 0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 6264.01  on 1000  degrees of freedom
Residual deviance:  867.47  on  991  degrees of freedom
AIC: 3610.3

Number of Fisher Scoring iterations: 5
```

We can see from the summary of the model that **intercept** , **Sealed**, **verifyID**, **LogBook** and **MinBidShare** are the most significant covariates.

Table 1: Maximum Likelihood Estimates

Const	PowerSeller	VerifyID	Sealed	Minblem	MajBlem	LargNeg	LogBook	MinBidShare
1.0724	-0.0205	-0.3945	0.4438	-0.0522	-0.2209	0.0707	-0.1207	-1.8941

2.2. Bayesian Analysis of the Poisson Regression

Prior: In this task we will do Bayesian analysis of the poisson regression. Let the prior be Zellner's g-prior:

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

log-Likelihood: Since we know from poisson model that,

$$p(y_i|\lambda) = \frac{e^{-\lambda}\lambda^{y_i}}{y_i!}$$

Like-lihood function:

$$L(y_i|\lambda) = \prod_{i=1}^{n} \frac{e^{-\lambda}\lambda^{y_i}}{y_i!}$$

$$= \prod_{i=1}^{n} \left(\frac{1}{y_i!}\right) * e^{-n\lambda}\lambda^{\sum_{i=1}^{n} y_i}$$

Taking log of the Likelihood function,

$$l(y_i|\lambda) = log\left(\prod_{i=1}^{n} \frac{e^{-\lambda}\lambda^{y_i}}{y_i!}\right)$$

$$= -n\lambda + \sum_{i=1}^{n} y_i * log(\lambda) - log\left(\sum_{i=1}^{n} y_i\right)$$
$$\propto -n\lambda + \sum_{i=1}^{n} y_i * log(\lambda)$$

replacing $\lambda = X_i^T \beta$ we get the following exoression:

$$l(y_i|\beta) \propto \sum_{i=1}^{n} y_i * log(X_i^T \beta) - \sum_{i=1}^{n} X_i^T \beta$$

Posterior Distribution of vector β

The 9-dim parameter vectoe β has the following distribution:

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

where $\tilde{\beta}$ is the posterior mode and $J(\tilde{\beta}) = -\frac{\partial^2 \ln\ p(\beta|y)}{\partial\beta\ \partial\beta^T}|_{\beta=\tilde{\beta}}$ is the negative of obsrved Hessian evaluated at the posterior mode.

Numerical Values for $\tilde{\beta}$:

Table 2: Betas

Const	PowerSeller	VerifyID	Sealed	Minblem	MajBlem	LargNeg	LogBook	MinBidShare
1.0698	-0.0205	-0.393	0.4436	-0.0525	-0.2212	0.0707	-0.1202	-1.892

The Matrix $J_y^{-1}(\tilde{\beta})$:

Table 3: Inverse of Hessian Matrix

0.00095	-0.00071	-0.00027	-0.00027	-0.00045	-0.00028	-0.00051	0.00006	0.00111
-0.00071	0.00135	0.00004	-0.00029	0.00011	-0.00021	0.00028	0.00012	-0.00057
-0.00027	0.00004	0.00852	-0.00078	-0.00010	0.00023	0.00033	-0.00032	-0.00043
-0.00027	-0.00029	-0.00078	0.00256	0.00036	0.00045	0.00034	-0.00013	-0.00006
-0.00045	0.00011	-0.00010	0.00036	0.00362	0.00035	0.00006	0.00006	-0.00006
-0.00028	-0.00021	0.00023	0.00045	0.00035	0.00837	0.00040	-0.00009	0.00026
-0.00051	0.00028	0.00033	0.00034	0.00006	0.00040	0.00318	-0.00025	-0.00011
0.00006	0.00012	-0.00032	-0.00013	0.00006	-0.00009	-0.00025	0.00084	0.00104
0.00111	-0.00057	-0.00043	-0.00006	-0.00006	0.00026	-0.00011	0.00104	0.00505

2.3. Posterior of β using MetroPolis Hasting Sampling

In this task we are simmulating from the actual posterior of β using the Metropolis Hasting algorithm and compare it with the approximated result in the previous step. We have generated a general function that random draws from an arbitrary posterior density consideringmultivariate normal density as the proposal density.

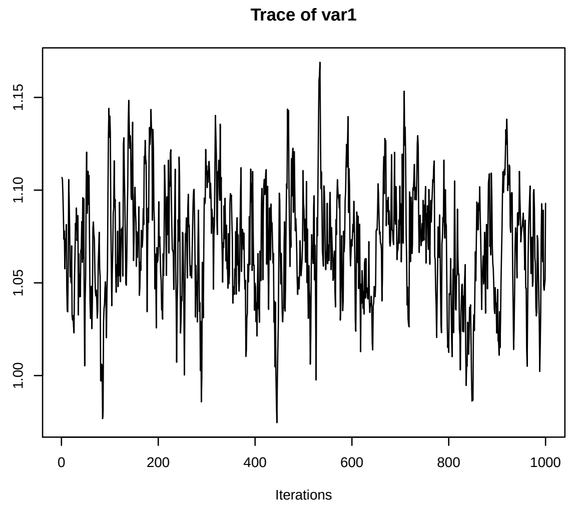
$$\theta_p|\theta^{i-1} \sim \mathcal{N}\big(\theta^{i-1}, c.\Sigma\big)$$

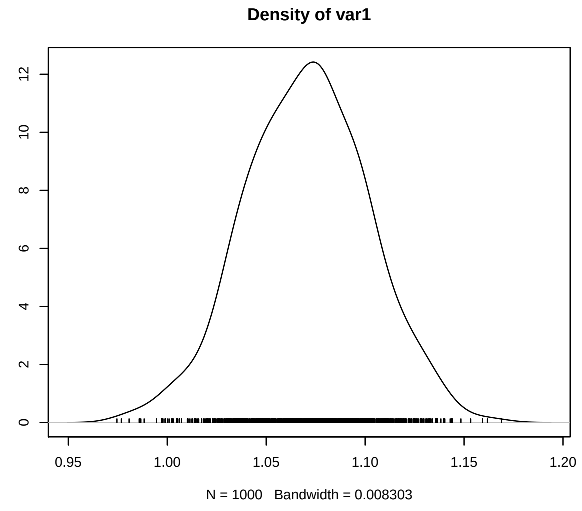
where $\Sigma = J_y^{-1}(\tilde{\beta})$ and c is the tuning parameter.

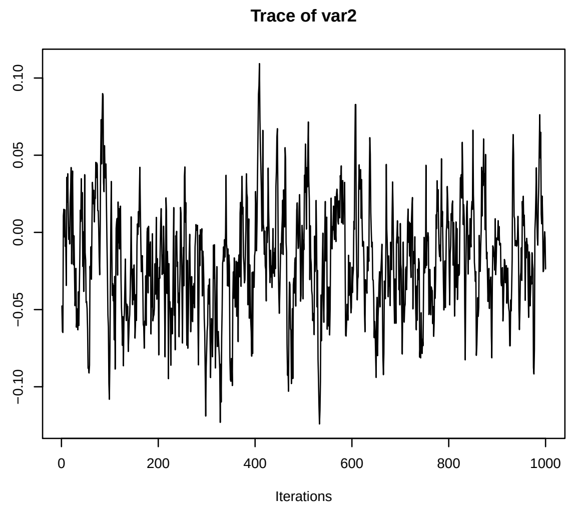
We have chosen the tuning parameter $c = 0.7$ as accptance rate of samplr for this parameter is 0.2828836%.

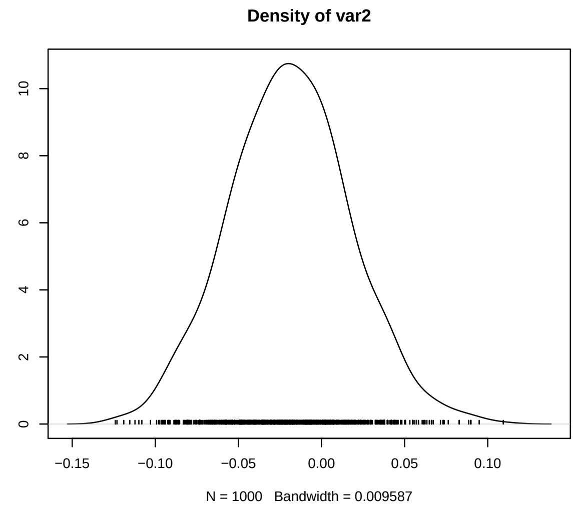
The following plots represents the sequential output from the objective distribution using Markov chain and density distribution of onjective function using markov chain.

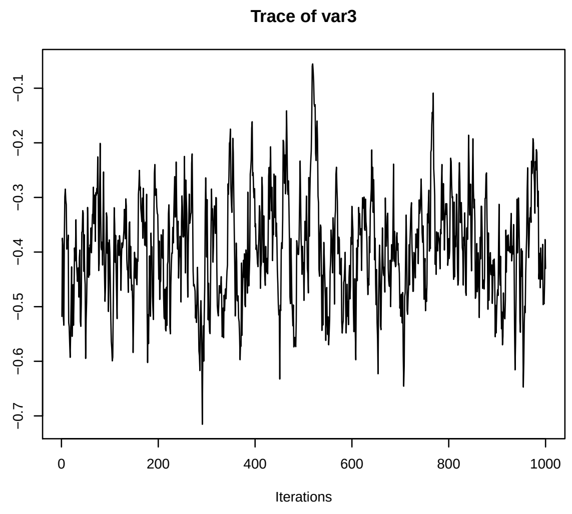
Trace plots of Variables

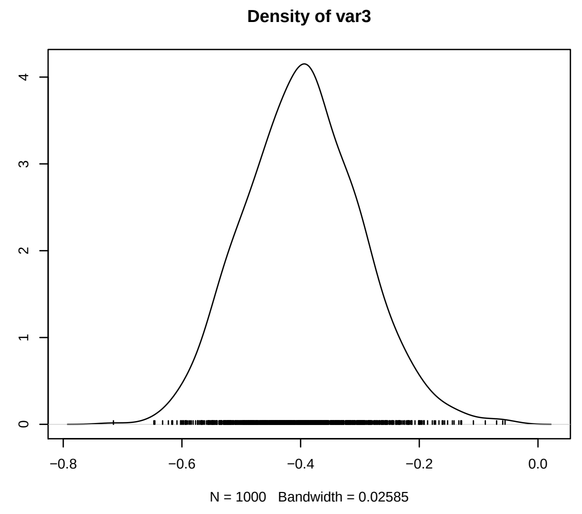


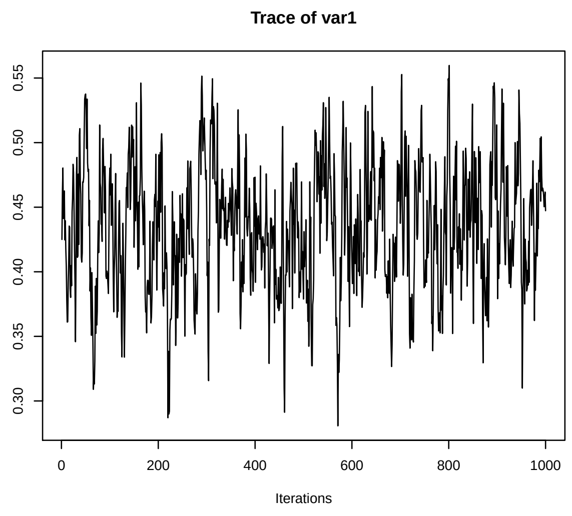


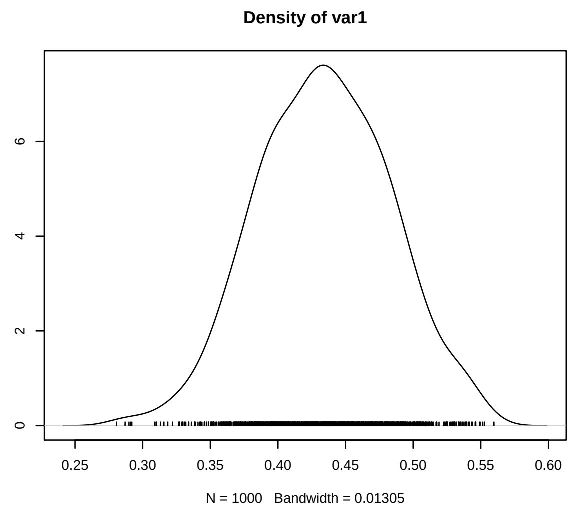


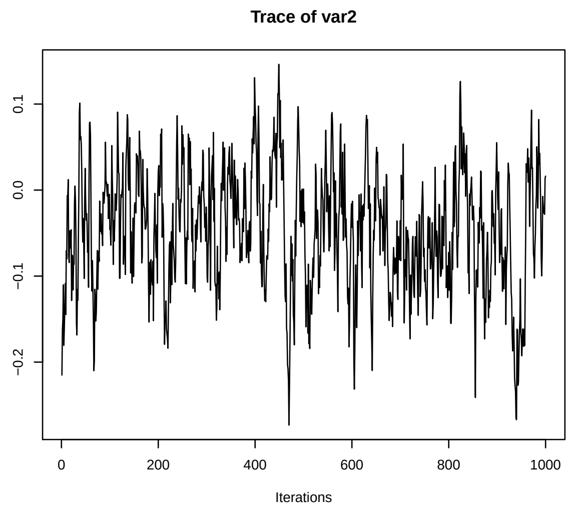


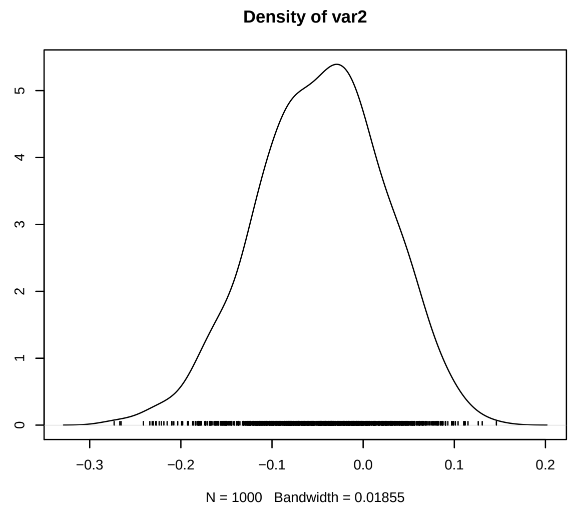


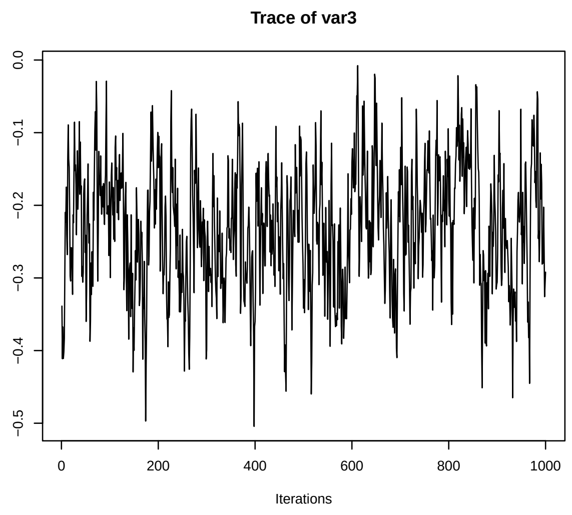


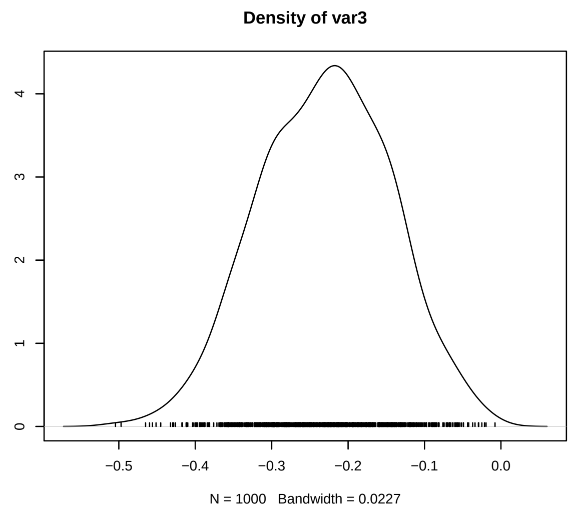

Trace of var1
Iterations

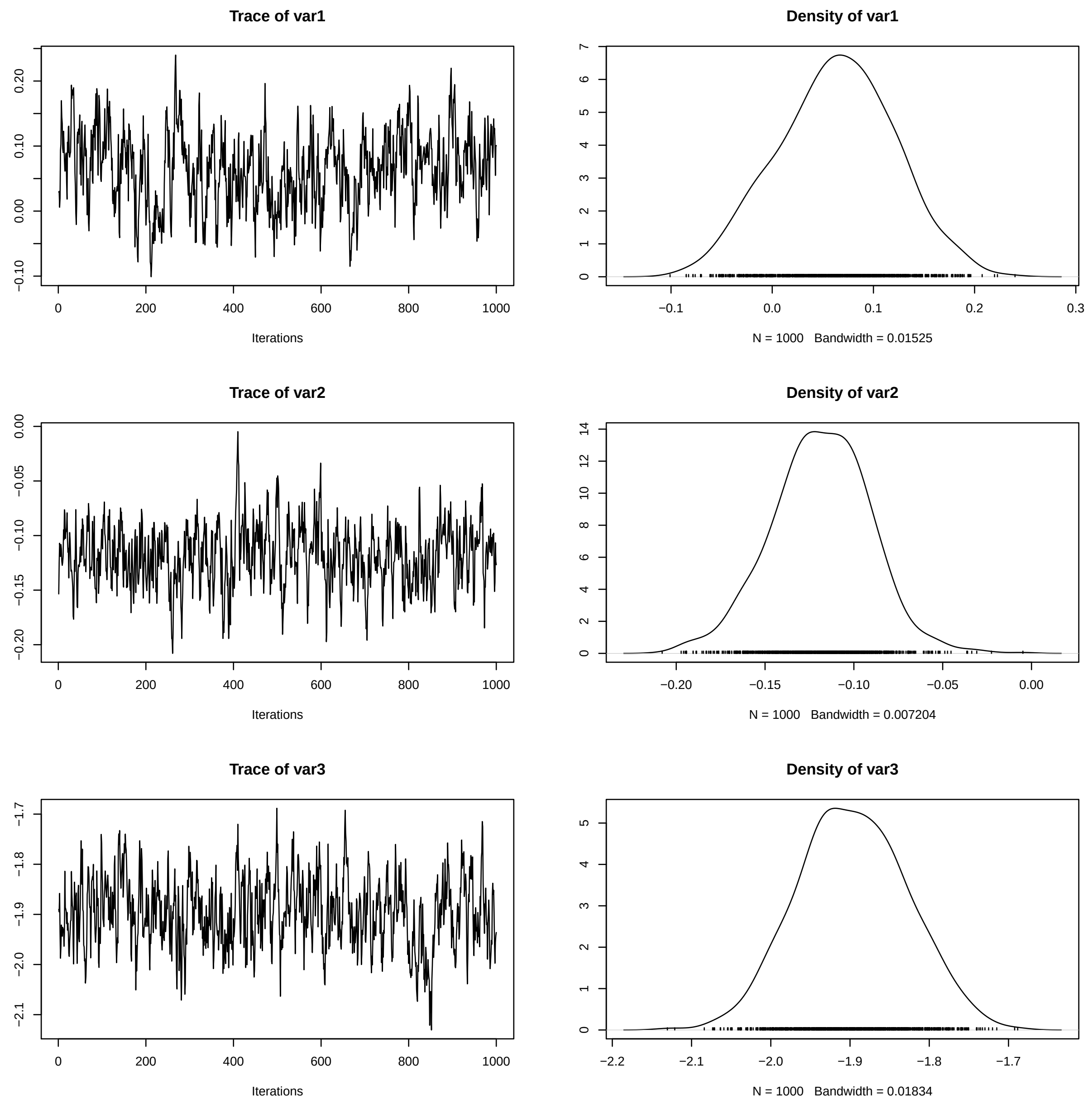

Density of var1
N = 1000 Bandwidth = 0.01305


Trace of var2
Iterations


Density of var2
N = 1000 Bandwidth = 0.01855


Trace of var3
Iterations


Density of var3
N = 1000 Bandwidth = 0.0227



In the above shown trajectories of the covariates we can see that markov chain is traversing in a similar fashion thus we can infer that it is showing convergence to a stationary distribution.
As we can see in all the time series plots that there is a quick descend or ascend of values from the starting point so we can say that there exist a noticeable burn-in period. Thus we should discard first half of the initial values to deminish the influence of the initial values.

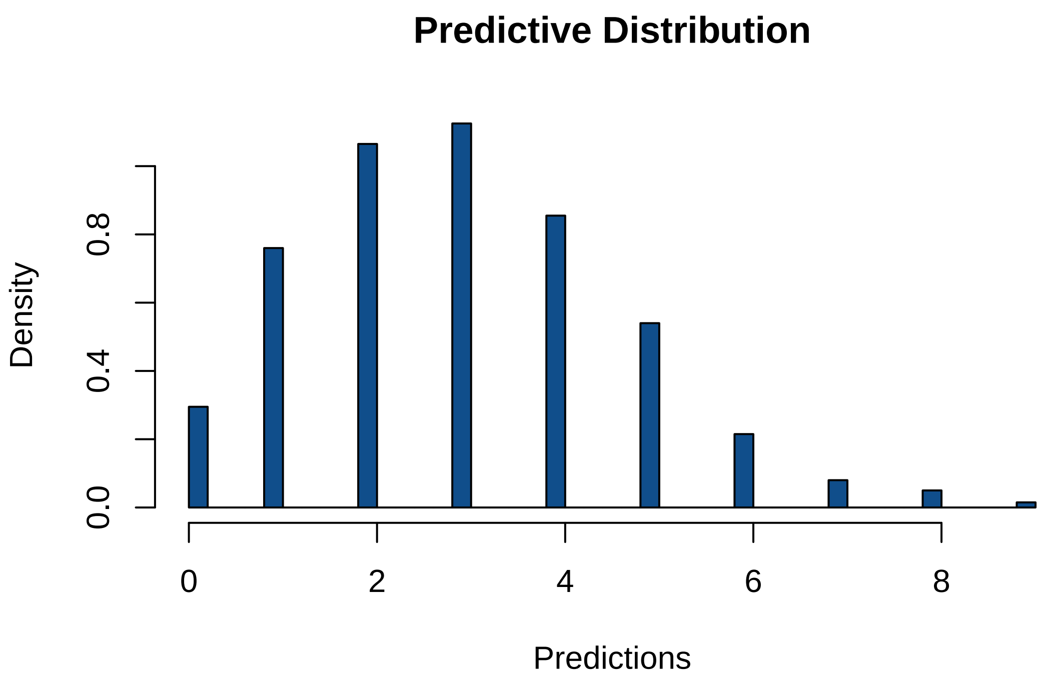
2.4. Predictive Distribution

In this task we are predicting response variable "nBids" in a new auction with the characteristics below:

Table 4: Data

Const	PowerSeller	VerifyID	Sealed	Minblem	MajBlem	LargNeg	LogBook	MinBidShare
1	1	1	1	0	0	0	1	0.5

```
Predictions
  0   1   2   3   4   5   6   7   8   9 
 59 152 213 225 171 108  43  16  10   3 
```



Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
library(coda)
library(knitr)
library(kableExtra)
library(mvtnorm)
library(latex2exp)
setwd("C:/Users/Rakshanda/Desktop/Bayesian Learning/LABS/lab_03")
options("scipen"=100)
data = read.csv("rainfall.dat")
#______(a)_______#
x = data$X136
hist(x, breaks = 50,probability = T, main = 'Histogram of RainFall Data')
lines(density(x),col='red',lwd = 1.5, main = 'Distribution of Data')
###### Defining a function that simulates from the
rInvchisq <- function(n, df, scale) (df*scale)/rchisq(n,df=df)
#___prior hyperparameters___#
prior = list()
prior$mu0 = 10
prior$tau2_0 = 50
prior$v0 = 10
prior$sig2_0 = 100
n =  length(x)
hist(x,breaks = 50,probability = T, ylim = c(0,0.04))
curve(expr = dnorm(x = x,mean = prior$mu0,sd = sqrt(prior$sig2_0)),
      lty=2,col = 'blue',add = T,lwd=2)
#' @param mu_0 prior mean of mu
#' @param tau2_0 prior SD of mu
#___Conditional distributions of mu___#
update_mu = function(nDraws,y,sigma2,tau2_0,mu0){
  n = length(y)
  tau2_n = (n/sigma2 + 1/tau2_0)^(-1)
  w = (n/sigma2)/(n/sigma2 + 1/tau2_0)
  mu_n = w*mean(y)+(1-w)*mu0
  mu = rnorm(1,mean = mu_n,sd = sqrt(tau2_n))
  return(mu)
}
#' @param sigma2_0 prior or best guess of sigma 2
#' @param v0 degrees of freedom for prior on sigma2
#___Conditional distributions of sigma2___#
update_sigma2 = function(nDraws,y,mu,v0,sig2_0){
  n = length(y)
  vn = v0 + n
  sig2_n = (v0*sig2_0 + sum((y-mu)^2)) / vn
  sigma2  = rInvchisq(n = 1,df = vn,scale = sig2_n)
  return(sigma2)
}
#' @param nDraws random Sample to draw from metropolis hasting sample
#' @param y observed data
#' @param theta_0 prior or best guess of the parameter
#' @param initial initial values for mu and sigma2
#' @param  prior a list of priors
```

```r
#___Posterior distribution using gibbs___#
gibbs =function(nDraws=1000,y,initial,prior){
  #browser()
  # initial values
  mu = initial[1]
  sigma2 = initial[2]

  draws = matrix(nrow = nDraws,ncol = 2)
  draws[1,] = c(mu,sigma2)
  for (i in 2:nDraws) {

    # update mu
    mu = update_mu(1,y,sigma2,tau2_0 = prior$tau2_0,mu0 = prior$mu0)
    draws[i,1] = mu

    # update sigma2
    sigma2  = update_sigma2(1,y,mu,v0 = prior$mu0,sig2_0 = prior$sig2_0)
    draws[i,2] = sigma2
  }
  draws
}
init_val = c(mean(x),var(x))
sample = gibbs(y = data$X136,initial = init_val,prior = prior)
mu_1 = sample[,1]
sigma2_1 = sample[,2]
par(mfrow = c(1,2))
hist(mu_1, xlab = 'Gibbs Sample',probability = T,
     main = TeX(paste("$\\mu |\\sigma^2,X$")),breaks = 30)
lines(density(mu_1), col ='red',lwd = 2)
hist(sigma2_1, xlab = 'Gibbs Sample',probability = T,
     main = TeX(paste("$\\sigma^2 |\\mu,X$")), breaks = 30)
lines(density(sigma2_1), col ='red',lwd = 2)
#  joint posterior of(mu,sigma2)
plot(x = mu_1[],y = sigma2_1, main = 'Joint Distribution',
     xlab = TeX("$\\mu$"), ylab = TeX("$\\sigma^2$"))
abline(h = var(x), v = mean(x), col = 'red')
ts.plot(mu_1,ylab = TeX(paste("$\\mu$")),
         main = TeX(paste("Trajectory of $\\mu$")))
ts.plot(sigma2_1,ylab = TeX(paste("$\\sigma^2$")),
        main = TeX(paste("Trajectory of $\\sigma^2$")))
#____Convergence using gelman rubin diagnostic____#
set.seed(12345)
init= c(0,1)
post1 = gibbs(nDraws = 1000,y = x,initial = init,prior)
init= c(5,0.01)
post2 = gibbs(nDraws = 1000,y = x,initial = init,prior)
init= c(10,0.5)
post3 = gibbs(nDraws = 1000,y = x,initial = init,prior)
init= c(20,0.4)
post4 = gibbs(nDraws = 1000,y = x,initial = init,prior)
init= c(15,10)
post5 = gibbs(nDraws = 1000,y = x,initial = init,prior)
```

```r
pmc_mu = mcmc.list(as.mcmc(post1[,1]),as.mcmc(post2[,1]),as.mcmc(post3[,1]),
                   as.mcmc(post4[,1]),as.mcmc(post5[,1]))

pmc_sig = mcmc.list(as.mcmc(post1[,2]),as.mcmc(post2[,2]),as.mcmc(post3[,2]),
                   as.mcmc(post4[,2]),as.mcmc(post5[,2]))

gelman.plot(pmc_mu, main = TeX(paste("Convergence of $\\mu$")))
gelman.diag(x = pmc_mu)
gelman.plot(pmc_sig, main = TeX(paste("Convergence of $\\sigma^2$")))
gelman.diag(x = pmc_sig)
# Prior options
prior = list()
prior$alpha = 7*rep(1,2)
prior$mu0 = rep(0,2)
prior$tau2_0 = rep(10,2)
prior$sigma2_0 = rep(var(x),2)
prior$v0 = rep(10,2)
#______(b)______#
#  a function that simulates from a Dirichlet distribution for pi
rDirichlet = function(alpha){
  n = length(alpha)
  x = rep(0,n)
  for (i in 1:n){
    x[i] = rgamma(1,alpha[i],1)
  }
  x = x/sum(x)
  return(x)
}

# Simple function that converts between two different representations of the mixture
# allocation.
Alloc = function(I){
  n = nrow(I)
  vect = rep(0,n)
  for (i in 1:n){
    vect[i] = which(I[i,] == 1)
  }
  return(vect)
}

#' @param nDraws number of samples
#' @param x given data
#' @param K number of components for mixture model
#' @param prior a list of hyperparameters of priors
#' @param prior$alpha Dirichlet(alpha)
#' @param prior$mu0 prior mean of mu
#' @param prior$tau2_0 prior SD of mu
#' @param prior$sigma2_0 prior or best guess of sigma 2
#' @param prior$v0 degrees of freedom for prior on sigma2

xGrid = seq(min(x)-1*apply(data,2,sd),max(x)+1*apply(data,2,sd),length = 100)

gibbs_mixture = function(nDraws=100,x,K=2,prior)
```

```r
{
  n = length(x)

  # Initial mcmc values
  I = t(rmultinom(n, size = 1 , prob = rep(1/K,K)))
  mu = rep(0,K)
  sigma2 = rep(var(x),K)
  probObsInComp = rep(NA,K)

  #variables to store final values
  MU = matrix(nrow = nDraws,ncol = K)
  Sigma2 = matrix(nrow = nDraws,ncol = K)

  #Computing mixture density means or thetas
  mixDensMean = rep(0,length(xGrid))
  count = 0

  for (it in 1:nDraws)
  {
    alloc = Alloc(I)
    nAlloc = colSums(I)

    # Update components probabilities
    pi = rDirichlet(prior$alpha + nAlloc)

    # Update mu's
    for (j in 1:K){
      precPrior = 1/prior$tau2_0[j]
      precData = nAlloc[j]/sigma2[j]
      precPost = precPrior + precData
      w = precPrior/precPost
      mu_n = w*prior$mu0 + (1-w)*mean(x[alloc == j])
      tau2_n = 1/precPost
      mu[j] = rnorm(1, mean = mu_n, sd = sqrt(tau2_n))
    }
    MU[it,] = mu

    # Update sigma2's
    for (j in 1:K){
      sigma2[j] = rInvchisq(1, df = prior$v0[j] + nAlloc[j],
                            scale = (prior$v0[j]*prior$sigma2_0[j] +
                                       sum((x[alloc == j] - mu[j])^2)) / (prior$v0[j] + nAlloc[j]))
    }
    Sigma2[it,] = sigma2

    # Update probability Observations in Computation
    for (i in 1:n){
      for (j in 1:K){
        probObsInComp[j] = pi[j]*dnorm(x[i], mean = mu[j], sd = sqrt(sigma2[j]))
      }
      I[i,] = t(rmultinom(1, size = 1 , prob = probObsInComp/sum(probObsInComp)))
    }
    # mixture of densities
    if (TRUE && (it%%1 ==0)){
      mixDens = rep(0,length(xGrid))
      count = count + 1
      for (j in 1:K){
        temp = pi[j] * dnorm(xGrid,mu[j],sd = sqrt(sigma2[j]))
        mixDens = mixDens + temp
      }
      mixDensMean = ((count-1)*mixDensMean + mixDens)/count
    }
  }

  lst = list(mu = MU, sigma2 = Sigma2, theta = mixDensMean)
  return(lst)
}
t = gibbs_mixture(x,nDraws = 500,K = 2,prior = prior)
mu = t$mu
sigma2 = t$sigma2
theta = t$theta
# conditional distributions of mu
par(mfrow=c(1,2))
hist(mu[,1], xlab = 'Gibbs Sample',probability = T,
     main = TeX(paste("$\\mu_1|I,\\sigma^2_1,X$")),breaks = 100)
lines(density(mu[,1]), col ='red',lwd = 2)

hist(mu[,2], xlab = 'Gibbs Sample',probability = T,xlim = c(10,20), ylim=c(0,1.2),
     main = TeX(paste("$\\mu_2|I,\\sigma^2_2,X$")),breaks = 100)
lines(density(mu[,2]), col ='red',lwd = 2)
par(mfrow=c(1,2))
hist(sigma2[,1], xlab = 'Gibbs Sample',probability = T,
     main = TeX(paste("$\\sigma^2_1|I,\\mu_1,X$")),breaks = 70)
lines(density(sigma2[,1]), col ='red',lwd = 2)

hist(sigma2[,2], xlab = 'Gibbs Sample',probability = T,,xlim = c(0,200), ylim=c(0,0.08),
  main = TeX(paste("$\\sigma^2_2|I,\\mu_2,X$")),breaks = 200)
lines(density(sigma2[,2]), col ='red',lwd = 1.5)
#___convergence plots___#
par(mfrow=c(2,1))
plot(mu[,1], type = 'l', ylim = c(0,70),ylab = TeX("$\\mu$"),
      main = TeX(paste("Trajectory of $\\mu=(\\mu_1,\\mu_2)$")))
lines(mu[,2], col = 'red')
legend("topright", box.lty = 1, bty = 'n',
       legend = TeX(c("$\\mu_1$","$\\mu_2$")), col=c("black","red"), lwd = 2)

plot(sigma2[,1], type = 'l',ylim = c(0,3000), ylab = TeX("$\\sigma^2$"),
     main = TeX(paste("Trajectory of $\\sigma^2=(\\sigma^2_1,\\sigma^2_2)$")))
lines(sigma2[,2], col = 'red')
legend("topright", box.lty = 1, bty = 'n', col=c("black","red"), lwd = 2,
       legend = TeX(c("$\\sigma_1^2$","$\\sigma_2^2$")))
hist(x, breaks = 50, freq = F,probability = T,
     xlim = c(min(xGrid),max(xGrid)),
     main = "Final Fitted Density")
lines(x = xGrid, y = theta, col = 'red', lwd = 2)
lines(xGrid,dnorm(xGrid, mean = mean(mu_1[500:1000]),sd = sqrt(mean(sigma2_1[500:1000]))),
      col = 'blue', lwd = 2, lty = 2)
legend('topright', legend = c("Histogram of Data","Guassian Mixture Model","Normal Model"),
       col=c("black","red","blue"), lwd = 1, bty = 'n', lty = c(1,1,2))
data = read.table("eBay.dat",header = T)
y = data$nBids
X = as.matrix(data[,-1])
#_______(a)______#
model = glm(formula = nBids ~0+.,data = data,family = "poisson")
summary(model)
MLE_coeff = matrix(round(model$coefficients,4),nrow = 1)
colnames(MLE_coeff) = colnames(X)
kable(MLE_coeff,"latex", caption = "Maximum Likelihood Estimates", booktabs = T) %>%
  kable_styling(latex_options = "HOLD_position")
#_____(b)______#
#' @param beta regression coefficients
#' @param y response variable
#' @param X features/independen variables
#' @param sigma prior or best guess of sigma 2

# prior
Sigma = 100 * solve(t(X)%*%X)
mu0 = rep(0,ncol(X))
log_posterior = function(beta,y,X,Sigma,mu0){
  beta = as.vector(beta)
  linPred = X%*%beta
  log_lik = sum(y * linPred) - sum(exp(linPred))
  if (abs(log_lik) == Inf) log_lik = -20000
  log_prior = dmvnorm(beta, matrix(c(mu0),length(beta),1), Sigma, log=TRUE)
  return(log_lik + log_prior)
}
optimal = optim(rep(0,9),log_posterior,gr=NULL,y,X,Sigma,mu0,method=c("BFGS"),
                control=list(fnscale=-1),hessian=TRUE)
optimal = optim(rep(0,9),log_posterior,gr=NULL,y,X,Sigma,mu0,
                method=c("BFGS"),control=list(fnscale=-1),hessian=TRUE)
betas = optimal$par
jInv = solve(-1*optimal$hessian)
beta = matrix(round(betas,4),nrow = 1)
colnames(beta) = colnames(X)
kable(beta,"latex", caption = 'Betas', booktabs = T) %>%
  kable_styling(latex_options = "HOLD_position")
kable(round(jInv,5),"latex", caption = "Inverse of Hessian Matrix", booktabs = T) %>%
 kable_styling(latex_options = "HOLD_position")
#______(c)_______#
#' @param postFun Log posterior of the objective distribution
#' @param nDraws random Sample to draw from metropolis hasting sample
#' @param y response variable
#' @param X features/independen variables
#' @param theta_0 prior or best guess of the parameter
#' @param c tuning parameter
MH = function(postFun, theta_0, c,iter,warmup,...)
{
  nDraws = iter + warmup
  optimal = optim(theta_0, postFun, gr=NULL,...,
                  method=c("BFGS"), control=list(fnscale=-1),
                  hessian=TRUE)
  jInv = solve(-optimal$hessian)*c
  X = matrix(nrow = nDraws, ncol = length(theta_0))
  X[1,] = theta_0
  accepted = 0
  count = 0
  for (i in 2:nDraws)
  {
    while(accepted < nDraws) {
     current = X[i-1,]
     theta_p = rmvnorm(1, mean = current, sigma = jInv)

    # acceptance probability ratio
    ratio = exp(postFun(c(theta_p),...) - postFun(c(current),...)) *
      dmvnorm(current,theta_p,jInv) / dmvnorm(theta_p,current,jInv)

    alpha =  min(1, ratio)
    u = runif(1)
    count = count + 1
    if(u<=alpha){
      X[i,] = theta_p
      current = theta_p
      accepted = accepted+1
      break
      }
    }
  }
  return(list(draws = X[-c(1:warmup),], acceptance_rate = accepted/count))
}
smp = MH(log_posterior,rep(0,9), c=0.6,iter = 1000, warmup = 500, y,X, Sigma,mu0)
betas = smp[[1]]
plot(as.mcmc(betas[,1:3]))
plot(as.mcmc(betas[,4:6]))
plot(as.mcmc(betas[,7:9]))
#___Prediction of the Response Variable___#
auction = matrix(c(1,1,1,1,0,0,0,1,0.5), nrow = 1)
colnames(auction) = colnames(X)
kable(auction, "latex",caption ='Data', booktabs = T) %>%
kable_styling(latex_options = "HOLD_position")
predictions = function(nDraws=1000, X = auction){
    draws = c()
    for (i in 1:nDraws) {
      probs = exp(X*betas[i,])
      draws[i] = rpois(n = 1,lambda = probs)
    }
    draws
}
Predictions = predictions()
table(Predictions)
hist(Predictions, breaks = 40, probability = T, col = "dodgerblue4",
     xlab = 'Predictions', main = 'Predictive Distribution')
```