732A91-Lab3-Report

Zhixuan Duan(zhidu838) Fengjuan Chen(fench417)

5/11/2020

1. Normal model, mixture of normal model

(a) Normal model

Assume the daily precipitation $\{y_1, ..., y_n\}$ are independent normally distributed, $y_1, ..., y_n|\mu, \sigma^2 \sim \mathcal{N}(\mu, \sigma^2)$ where both μ and σ^2 are unknown.

Let the priors of μ and σ^2 are $\mu \sim \mathcal{N}(\mu_0, \tau_0^2)$ and $\sigma^2 \sim Inv - \chi^2(v_0, \sigma_0^2)$ respectively.

We want to use the Gibbs sampling algorthim to simulate draws from the joint posterior distribution $p(\mu, \sigma^2|y_1, ..., y_n)$.

The Gibbs sampling algorithm for a multivariate distribution, $p(\theta_1, \theta_2, ..., \theta_k)$ is described below.

Setp 1. Choose initial values $\theta_2^{(0)}, \theta_3^{(0)}, ..., \theta_k^{(0)}$

Step 2. Repeat for $j = 1, ..., N$:

Draw $\theta_1^{(j)}$ from $p(\theta_1|\theta_2^{(j-1)}, \theta_3^{(j-1)}, ..., \theta_k^{(j-1)})$

Draw $\theta_2^{(j)}$ from $p(\theta_2|\theta_1^{(j)}, \theta_3^{(j-1)}, ..., \theta_k^{(j-1)})$

...

Draw $\theta_k^{(j)}$ from $p(\theta_k|\theta_1^{(j)}, \theta_2^{(j)}, ..., \theta_{k-1}^{(j)})$

Step 3. Return draws: $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(N)}$, where $\theta^{(j)} = \theta_1^{(j)}, ..., \theta_k^{(j)}$.

(i) Implement a Gibbs sampler

To implement a Gibbs sampler that simulates from the joint posterior $p(\mu, \sigma^2|y_1, ..., y_n)$, we need the full conditional posteriors $\mu|\sigma^2, y \sim$ and $\sigma^2|\mu, y$ which are

$\mu|\sigma^2, y \sim \mathcal{N}(\mu_n, \tau_n^2)$ and $\sigma^2|\mu, y \sim Inv - \chi^2(v_n, \frac{v_0\sigma_0^2 + \sum_{i=1}^n (y_i - u)^2}{n + v_0})$

where $\frac{1}{\tau_n^2} = \frac{n}{\sigma^2} + \frac{1}{\tau_0^2}$, $\mu_n = w \cdot \bar{y} + (1 - w) \cdot \mu_0$, $w = \frac{\frac{n}{\sigma^2}}{\frac{n}{\sigma^2} + \frac{1}{\tau_0^2}}$, $v_n = v_0 + n$, $v_n\sigma_n^2 = v_0\sigma_0^2 + \sum_{i=1}^n (y_i - u)^2$,

n is the number of observations, $\bar{y}$ is the mean of all observations and y_i is the ith observation.

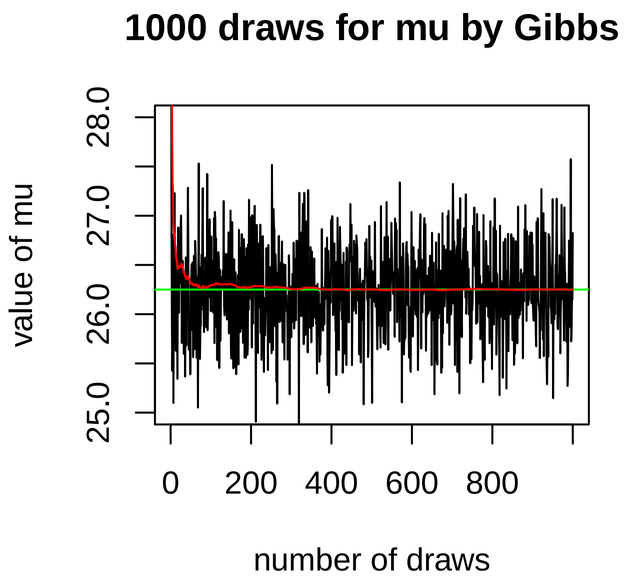
Here, we only have two parameters, μ and σ^2. So, we choose an initial value of σ^2 ($\sigma^2 = 1$) and iteratively draw from $\mu|\sigma^2, y$ and $\sigma^2|\mu, y$ with the updated values of μ and σ^2. (The hyperparameters for priors are set to $\mu_0 = 0$,$\tau_0^2 = 1$, $v_0 = 1$, $\sigma_0^2 = 1$)

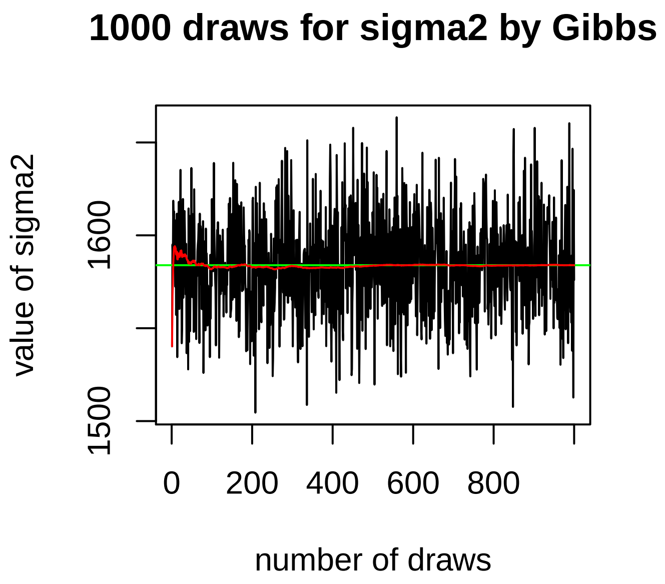
The code is shown in Appendix.

(ii) Analyze the daily precipitation using Gibbs sampler

We simulate 1000 draws from the joint posterior distribution $p(\mu, \sigma^2|y_1, ..., y_n)$ by Gibbs sampling algorithm. From the trace plots below, we can find that these 1000 draws are converged and they have little autocorrelation. The traceplots are very similiar with the plot of iid observations and this shows the simulation of these 1000 draws is good. We plot the cumulative mean for each parameter as a red line in the traceplot and the expected value as a green line. From these two lines, we can also conclude that the Markov chains are converged.

We also use some functions from "coda" package to analyze the autocorrelation and effecitvesize of this Markov chain. It confirms our conclusion. The codes are included in Appendix.





```
## The expected value of each parameter in 1000 draws: 26.25024 1583.915
```

(b) Mixture normal model

Now we assume the daily precipitation $\{y_1, ..., y_n\}$ follow an iid two-component mixture of normals model: $p(y_i|\mu, \sigma^2, \pi) = \pi\mathcal{N}(y_i|\mu_1, \sigma_1^2) + (1-\pi)\mathcal{N}(y_i|\mu_2, \sigma_2^2)$, where $\mu = (\mu_1, \mu_2)$, and $\sigma^2 = (\sigma_1^2, \sigma_2^2)$.

Since we have two components, the prior of π is $Beta(\alpha_1, \alpha 2)$, and the posterior of π is $Beta(\alpha_1 + n_1,\ \alpha 2 + n_2)$, where n_1 is the number of observations belonging to the first component while n_2 is the number of observations belonging to the second component. And we have $n_1 + n_2 = n$.

The prior and posterior of μ and σ^2 are same with 1 (a).

To use the Gibbs sampling data augmentation algorithm, we add an indicator $I_i = 0$ or $I_i = 1$ to each observation to show which distribution this observation belongs to.

We first set the indicator I of the values 0 and 1 with equal probability. Then we simulate the π according to I and data.

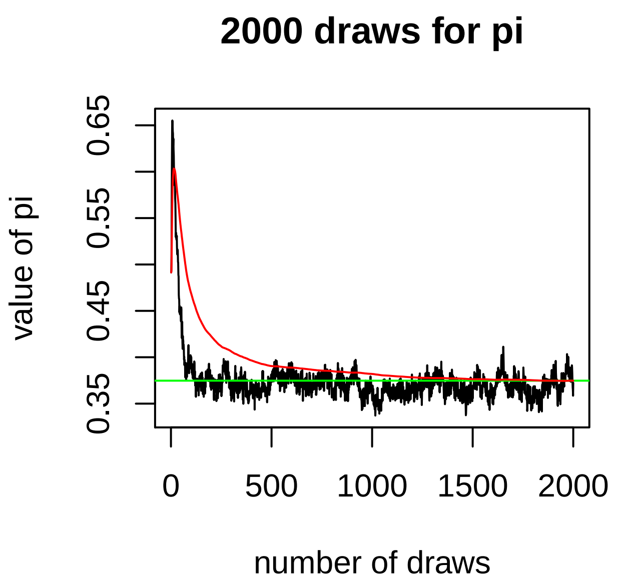
After that, we simulate μ and σ^2 similar as 1(a). But for μ_1 and σ_1^2, we only use observations belonging to the first normal distribution according to the value of indicator I_i. The same strategy for simulating μ_2 and σ_2^2.

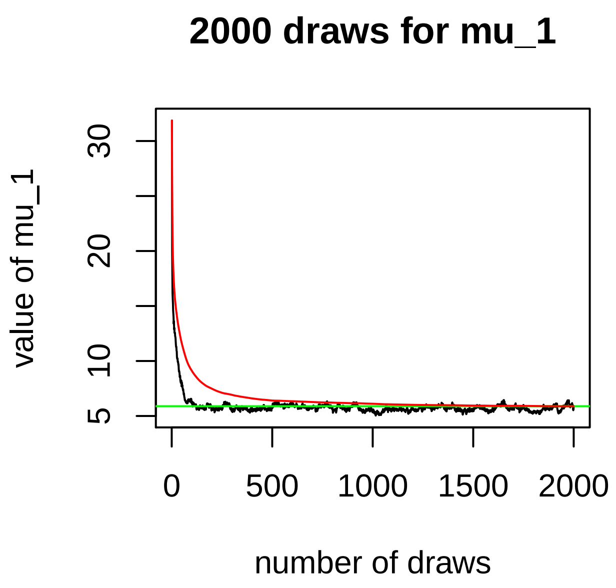
Then, we calculate the probability of each I_i corresponding to the observation x_i by using the formula $\theta_i = \frac{\pi\cdot\Phi(x_i;\mu_1,\sigma_1^2)}{\pi\cdot\Phi(x_i;\mu_1,\sigma_1^2)+(1-\pi)\cdot\Phi(x_i;\mu_2,\sigma_2^2)}$ and update the indicator vector I and enter to the next simulting iteration.

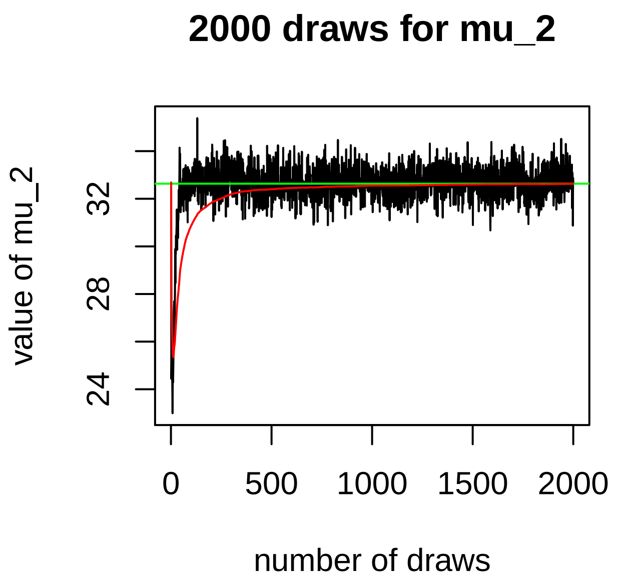
To generate the the indicator I, we also use the rmultinom() function just like the given code in NormalMixtureGibbs.R. We use two columns consisting of integers 0 and 1 to represent the first normal distribution and the second normal distribution respectively. The probabilities of generating them by rmultinom() are θ_i and $1 - \theta_i$.

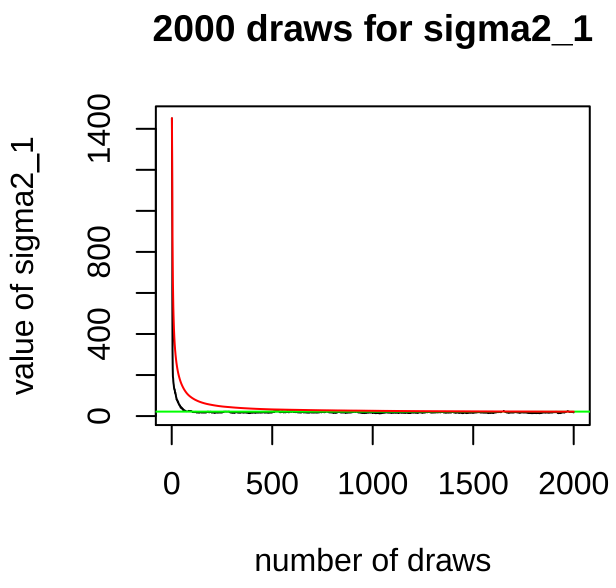
The prior hyperparameters are as follows: $\alpha_1 = 2$,$\alpha_2 = 2$, $\mu_0 = (0, 0)$, $\tau_0^2 = (1, 1)$, $v_0 = (1, 1)$, $\sigma 2_0 = (1, 1)$.

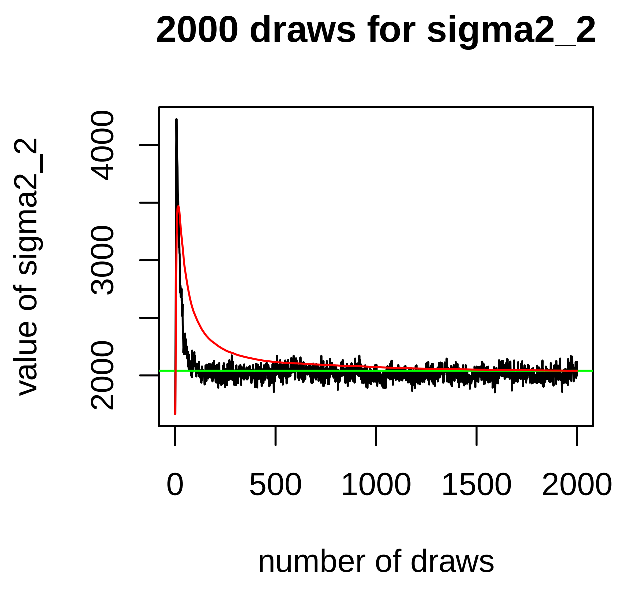
From the trace plots we can see that some parameters are not converged well for 1000 draws. But all of them are converged from 1000 draws to 2000 draws.









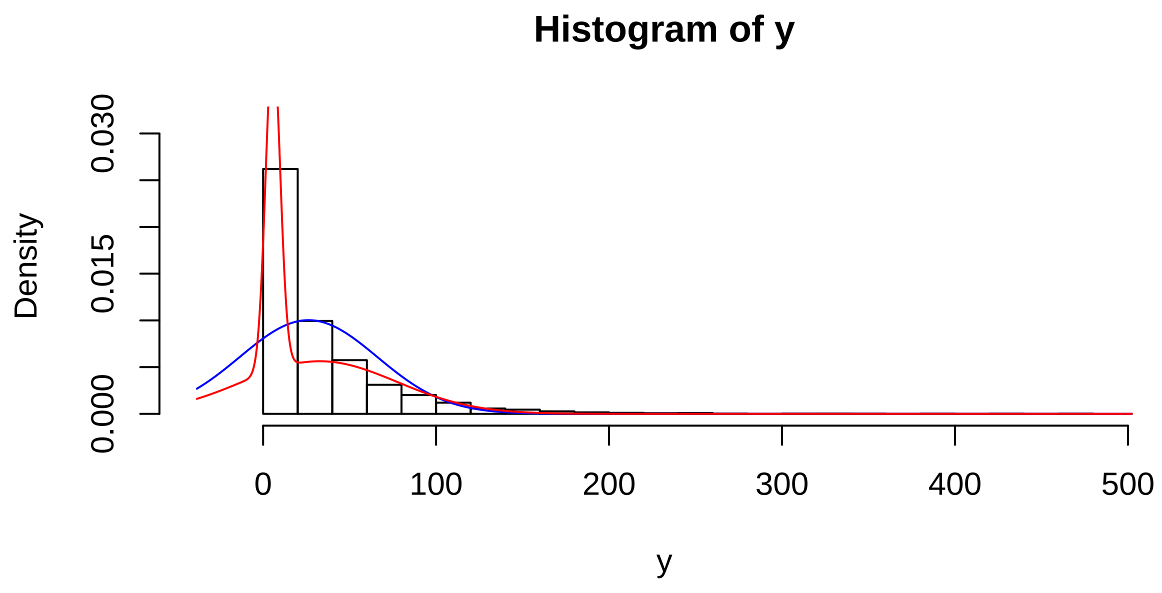


```
## The expected value of each parameter in 2000 draws: 0.3747338 5.886709 32.63302 21.63708 2040.885
```

(c) Graphical comparison

We use the mean of 1000 draws for mu and σ^2 obtained in (a) as the final values of mu and σ^2 in $\mathcal{N}(y_i|\mu,\ \sigma^2)$. For the mixture model, we use the mean of 1000 draws after burn-in period for 5 parameters,$\{\pi,\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\}$ as the final values to achieve the mixture normal model.

We plot the histogram of the data, noraml density $\mathcal{N}(y_i|\mu,\ \sigma^2)$ in (a) and mixture of normals density $p(y_i|\mu,\sigma^2,\pi) = \pi\mathcal{N}(y_i|\mu_1,\sigma_1^2) + (1-\pi)\mathcal{N}(y_i|\mu_2,\sigma_2^2)$ in (b) on one figure. The normal density in (a) is represented by blue line while the mixture normal density in (b) is illustrated by red line.



2. Metropolis Random Walk for Poinsson regression.

(a) Use glm to obtain the MLE esitamtor of beta

We use glm() function to obtain the MLE estimator of β in the poisson regression model $y_i|\beta \sim Poisson[exp(\boldsymbol{X_i}^T\beta)]$ for the eBay data.

```r
glmModel=glm(nBids~0+., data = eBay, family = poisson)
summary(glmModel)
```

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

From the results of the glm(), we can see that 4 covariates (VerifyID, Sealed, LogBook, MinbidShare) are significant because their p-values are less than $2e-16$.

(b) Approximate posterior density of beta

Now we do a Bayesian analysis of the Poisson regression.

We assume that the posterior density is approximately multivariate normal: $\beta|y \sim \mathcal{N}(\tilde{\beta}, J_y^{-1}(\tilde{\beta}))$, where $\tilde{\beta}$ is the posterior mode and $J_y^{-1}(\tilde{\beta})$ is the negative Hessian at the posteiror mode.

We still use the numerical optimization (optim.R) which we used in lab2 to obtain the values of $\tilde{\beta}$ and $J_y^{-1}(\tilde{\beta})$. Therefore we need to construct the expression proportional to $logp(\theta|y)$, which is $log\_likelihood + log\_Prior$.

Because this is Poisson regression model, we have $Pr(y_i|\beta) = \frac{\lambda^{y_i} \cdot e^{-\lambda}}{y_i!} = \frac{[exp(x_i^T\beta)]^{y_i} \cdot e^{-exp(x_i^T\beta)}}{y_i!}$.

Then the likelihood is $p(y|X,\beta) = \prod_{i=1}^n \frac{[exp(x_i^T\beta)]^{y_i} \cdot e^{-exp(x_i^T\beta)}}{y_i!}$.
The log_likelihood is

$logp(y|X,\beta) = log\{\prod_{i=1}^n \frac{[exp(x_i^T\beta)]^{y_i} \cdot e^{-exp(x_i^T\beta)}}{y_i!}\}$
$= log\sum_{i=1}^n[exp(x_i^T\beta)]^{y_i} + log\sum_{i=1}^n[e^{-exp(x_i^T\beta)}] - log(\prod_{i=1}^n y_i!)$
$= \sum_{i=1}^n[y_i \cdot x_i^T\beta - e^{x_i^T\beta} - ln(y_i!)]$
The Prior is $\beta \sim \mathcal{N}(0, 100 \cdot (X^TX)^{-1})$, where X is the $n \times p$ covariate matrix.

So, the log_Prior is the log of density of $\beta \sim N(0, 100 \cdot (X^TX)^{-1})$, which can be obtained by

dmvnorm(beta, mean = mu, Sigma, log=TRUE) where $\mu = 0$, $\Sigma = 100 \cdot (X^TX)^{-1}$.

We set the initial values of β as $init\_beta = (0,0,0,0,0,0,0,0,0)$.

The code is shown in Appendix.

After optim() function, we got the $\tilde{\beta}$ and $J_y^{-1}(\tilde{\beta})$.

Table 1: The values of tilde_beta

Const	PowerSeller	VerifyID	Sealed	Minblem	MajBlem	LargNeg	LogBook	MinBidShare
1.069841	-0.0205125	-0.393006	0.4435555	-0.0524663	-0.2212384	0.0706968	-0.1202177	-1.891985

We can see that the values of $\tilde{\beta}$ is very close to the values for the corresponding parameters obtained from glm() model.

(c) Random Walk Metropolis algorithm for arbitrary posterior density

The steps of Random Walk Metropolis algorithm:

Step 1. Initialize $\theta^{(0)}$.

Step 2. Repeate for i=1,2,...

Step 2(i). Sample a draw from the proposal distribution: $\theta_p|\theta^{(i-1)} \sim \mathcal{N}(\theta^{(i-1)}, c \cdot \Sigma)$.

Step 2(ii). Compute the acceptance probability $\alpha = min(1, \frac{p(\theta_p|y)}{p(\theta^{(i-1)}|y)})$.

Step 2(iii). With probability α accept the proposed value, and set $\theta^{(i)} = \theta_p$. Otherwise set $\theta^{(i)} = \theta^{(i-1)}$.

In step 2(ii), the $p(\theta_p|y)$ and $p(\theta^{(i-1)}|y)$ are the posterior distribution with different values of parameter θ.

In step 2(iii), with probability α accept the proposed value means if we generate a random number $u \sim Unif(0,1)$ and $u < \alpha$ then we accept the proposed value, otherwise we reject it.

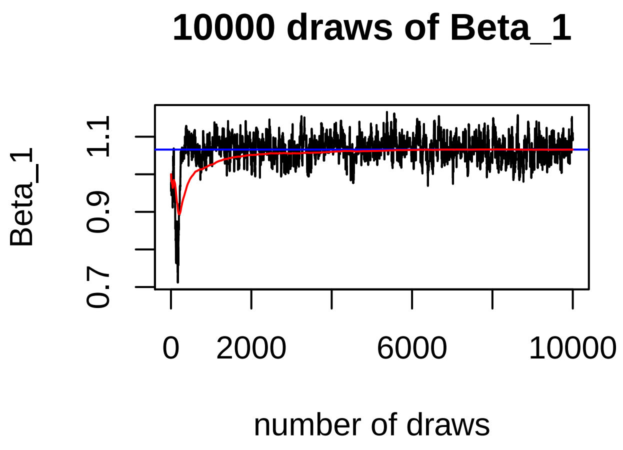
In our problem, we have the proposal density $\theta_p|\theta^{(i-1)} \sim \mathcal{N}(\theta^{(i-1)}, c \cdot \Sigma)$ where $\Sigma = J_y^{-1}(\tilde{\beta})$.

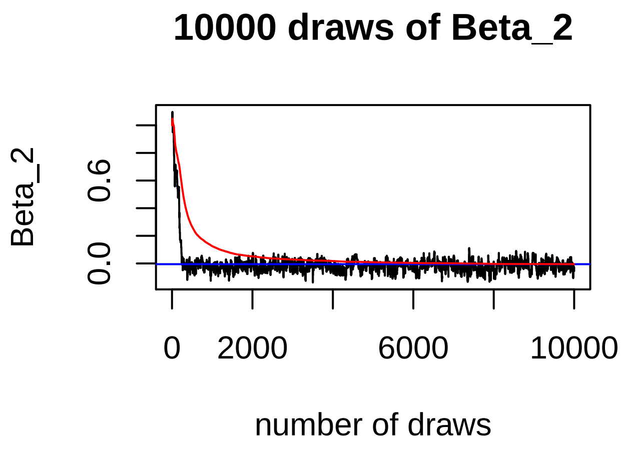
We have accomplished the log posterior function in 2 (b) above when we use optim() function to calculate the $\tilde{\beta}$ and $J_y^{-1}(\tilde{\beta})$.

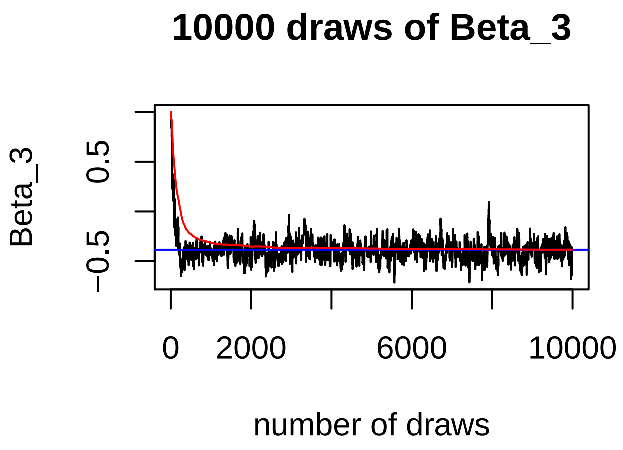
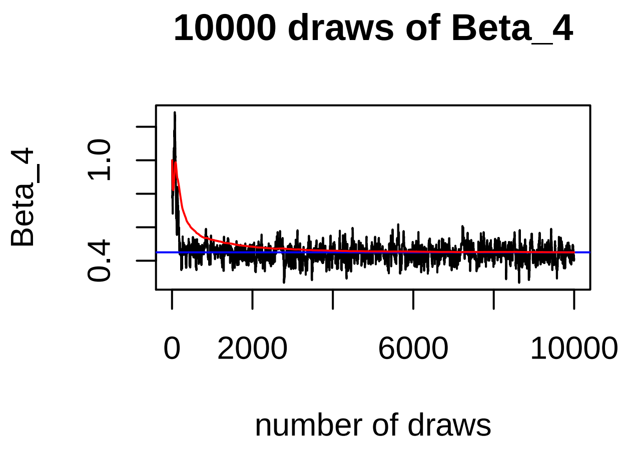
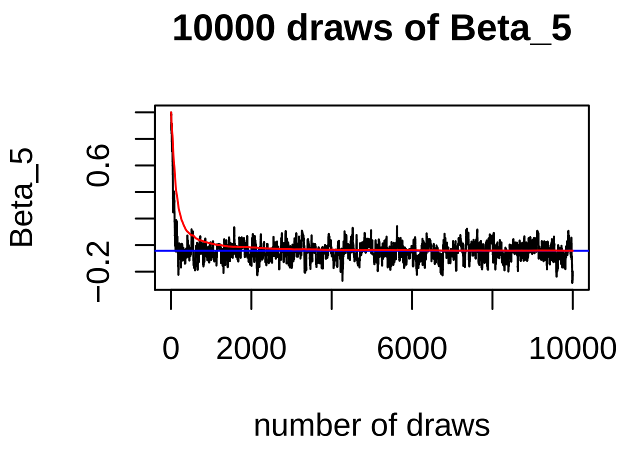
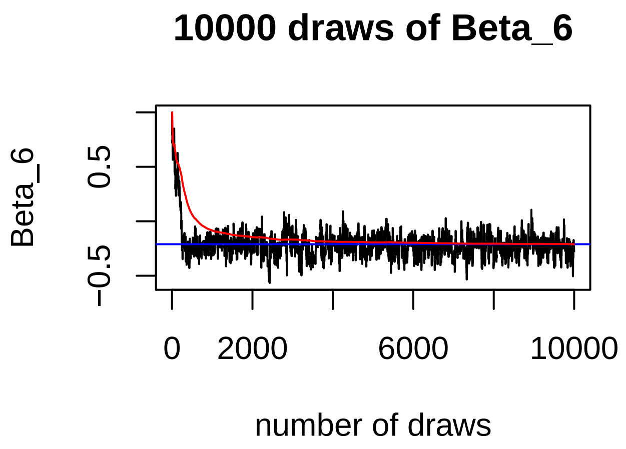
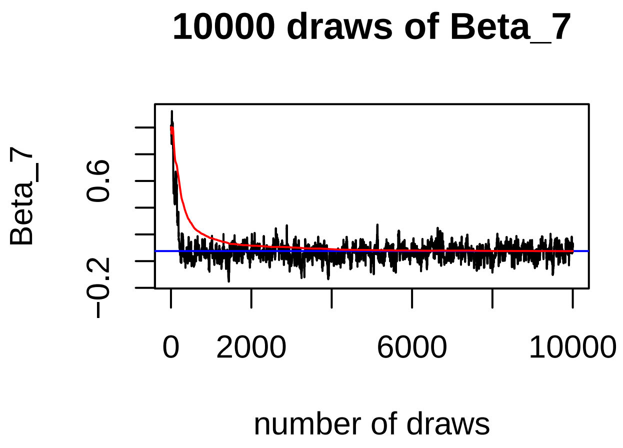
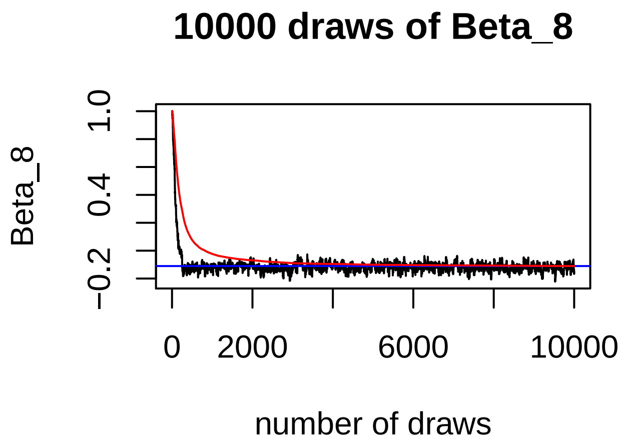
The parameter c is a tuning parameter. Tuning it to satisfy the average acceptance probability is 25-30%.

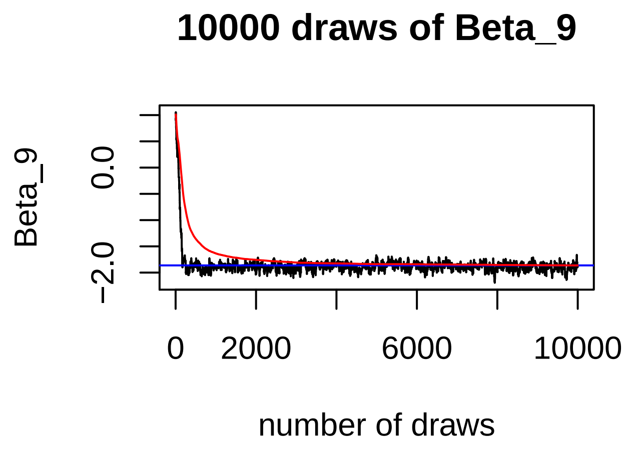
We use the Metropolist function to sample 10000 draws from the posterior of β in the Poisson regression for the eBay dataset and plot the trace plots for all paramters below.

```
## The acceptance rate is  0.2794
```






10000 draws of Beta_3
Beta_3
0.5
−0.5
0
2000
6000
10000
number of draws

10000 draws of Beta_4
Beta_4
1.0
0.4
0
2000
6000
10000
number of draws

10000 draws of Beta_5
Beta_5
0.6
−0.2
0
2000
6000
10000
number of draws

10000 draws of Beta_6
Beta_6
0.5
−0.5
0
2000
6000
10000
number of draws

10000 draws of Beta_7
Beta_7
0.6
−0.2
0
2000
6000
10000
number of draws

10000 draws of Beta_8
Beta_8
1.0
0.4
−0.2
0
2000
6000
10000
number of draws



From the trace plots, we find that the burn in period is about 3000 draws for all the elements of β.

(d) Predictive distribution of the number of bidders

In this section we use the MCMC draws from 2(c) to simulate from the predictive distribution of the number of bidders in a new auction with the X=c(1,1,1,1,0,0,0,1,0.5).

For each draw of β vector, we use this β vector to calculate the probability for each count of y (from 0 to 12). Then we set the our predicted count as the count with the highest probability. At last, we obtain all the simulated predictive numbers of bidders for this new auction.

To obtain better predicted results, we only want to use the draws from MCMC after burn-in period. Therefore, we truncate our chain from 3001 to 10000. Using these draws of *Beta*, we obtain the predictive distribution and plot it below. And we also illustrate the probability of each count of y in a table.

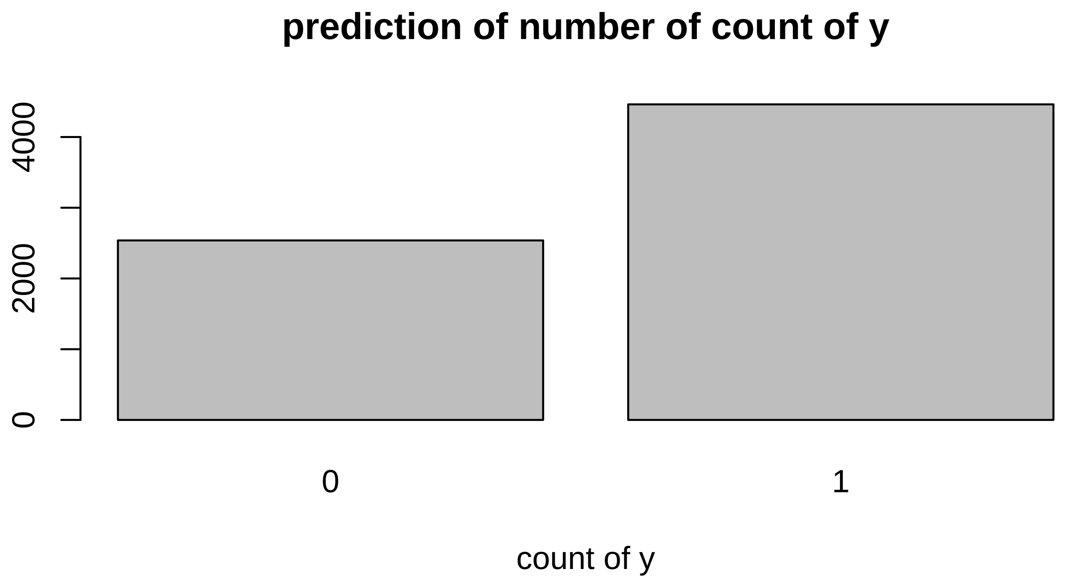


Table 2: the probability for each count(without burn in)

Num	0	1	2	3	4	5	6	7	8	9	10	11	12
count	2538	4462	0	0	0	0	0	0	0	0	0	0	0
prob	0.362571428571429	0.637428571428571	0	0	0	0	0	0	0	0	0	0	0

```
## The probability of no bidders in this new auction is (without burn-in) 0.3625714
```

Appendix

```r
# 1. Normal model, mixture of nomal model
rainfall=read.table("rainfall.dat",header = TRUE)
library(mvtnorm)
library(coda)
## (a) Normal model
gibbs <- function(nDraw){
  mu_0=0
  tau2_0=1
  v_0=1
  y=rainfall$X136
  sigma2_0=1
  n=nrow(rainfall)
  y_bar=mean(y)
  # choose an initial value of sigma2
  sigma2=1
  gib_sam=matrix(0,nrow = nDraw,ncol = 2)
  # cumulative results of two parameters
  cum_par=matrix(0,nrow = nDraw,ncol = 2)
  for (i in 1:nDraw) {
    # simulate mu from sigma2,y
    w=(n/sigma2)/(n/sigma2+1/tau2_0)
    mu_n=w*y_bar+(1-w)*mu_0
    tau2_n=1/(n/sigma2+1/tau2_0)
    mu=rnorm(1,mean=mu_n,sd=sqrt(tau2_n))
    gib_sam[i,1]=mu
    # simulate sigma2 from u, y
    v_n=v_0+n
    chi=rchisq(1,df=v_n)
    sigma2=(v_0*sigma2_0+sum((y-mu)^2))/chi
    gib_sam[i,2]=sigma2
    #cum_par[i,]=colMeans(gib_sam[1:i,])
    cum_par[i,]=c(mean(gib_sam[1:i,1]),mean(gib_sam[1:i,2]))
  }
  plot(c(1:nDraw),gib_sam[1:nDraw,1],type = "l",
       xlab = "number of draws",ylab = "value of mu",
       ylim = c(25.5,27),
       main = paste(nDraw,"draws for mu by Gibbs",sep = " "))
  abline(h=mean(gib_sam[,1]),col="green")
  lines(1:nDraw, cum_par[,1],col="red")
  plot(c(1:nDraw),gib_sam[1:nDraw,2],type="l",
       xlab = "number of draws", ylab = "value of sigma2",
       main = paste(nDraw,"draws for sigma2 by Gibbs",sep = " "))
  abline(h=mean(gib_sam[,2]),col="green")
  lines(1:nDraw, cum_par[,2],col="red")
  cat(nDraw, "draws", "means of each columns:",colMeans(gib_sam))
  return(gib_sam)
}

gib_sam=gibbs(nDraw = 1000)
## check convengence by "coda" package
effectiveSize(as.mcmc(gib_sam[,1]))
effectiveSize(as.mcmc(gib_sam[,2]))
autocorr.plot(as.mcmc(gib_sam[,1]))
autocorr.plot(as.mcmc(gib_sam[,2]))

## (b) Mixture normal model
mix_gib <- function(nDraw){
  n_comp=2
  mu_0=rep(0,n_comp)
  tau2_0=rep(1,n_comp)
  v_0=rep(1,n_comp)
  y=rainfall$X136
  sigma2_0=rep(1,n_comp)
  n=nrow(rainfall)
  alpha_1=2
  alpha_2=2
  # set initial value of indicator I
  indicator=t(rmultinom(n,1,prob = rep(1/n_comp,n_comp)))
  # choose an initial value of sigma2
  sigma2=rep(1,n_comp)
  # 5 parameters need to store, pi, mu1, sigma2_1, mu2, sigma2_2
  res=matrix(0,nrow = nDraw,ncol = 5)
  cum_par=matrix(0,nrow = nDraw,ncol = 5)
  mu=rep(0,n_comp)
  for (j in 1:nDraw) {
    # simulate pi according to indicator I
    ## post_pi=beta(alpha1+n1,alpha2+n2)
    ind_n=colSums(indicator)
    pi=rbeta(1,alpha_1+ind_n[1],alpha_2+ind_n[2])
    # simulate mu and sigma2
    ## simulate mu from sigma2,y and indicator I
    w=(ind_n/sigma2)/(ind_n/sigma2+1/tau2_0)
    y_bar=c(mean(y[which(indicator[,1]==1)]),
            mean(y[which(indicator[,2]==1)]))
    mu_n=w*y_bar+(1-w)*mu_0
    tau2_n=1/(ind_n/sigma2+1/tau2_0)
    for (i in 1:n_comp) {
      mu[i]=rnorm(1,mean=mu_n[i],sd=sqrt(tau2_n[i]))
    }
    ## simulate sigma2 from mu, y and indicator I
    v_n=v_0+ind_n
    for (i in 1:n_comp) {
      chi=rchisq(1,df=v_n[i])
      y_i=y[which(indicator[,i]==1)]
      sigma2[i]=(v_0[i]*sigma2_0[i]+sum((y_i-mu[i])^2))/chi
    }
# simulate indicator I from pi, mu and sigma2 for each observation
    for (i in 1:n) {
      val_comp1=dnorm(y[i],mean = mu[1],sd=sqrt(sigma2[1]))
      val_comp2=dnorm(y[i],mean = mu[2],sd=sqrt(sigma2[2]))
      theta_i=pi*val_comp1/(pi*val_comp1+(1-pi)*val_comp2)
      indicator[i,]=t(rmultinom(1,size=1,prob = c(theta_i,1- theta_i)))
    }
    # store the parameters obtained from this draw
    ## store the first parameter pi for the jth draw
    res[j,1]=pi
    ## store the second and third parameters mu1, mu2
    res[j,2:3]=mu
    ## store the fourth and fifth parameters sigma2_1, sigma2_2
    res[j,4:5]=sigma2
    cum_par[j,]=c(mean(res[1:j,1]),mean(res[1:j,2]),
                  mean(res[1:j,3]),mean(res[1:j,4]),
                  mean(res[1:j,5]))
  }

  show_name=c("pi","mu_1","mu_2","sigma2_1","sigma2_2")
  for (i in 1:ncol(res)) {
    plot(c(1:nDraw),res[,i],
         type = "l", xlab = "number of draws",
         ylab = paste("value of",show_name[i],sep = " " ),
         main = paste(nDraw,"draws for",show_name[i],
                      "of mixture model",sep = " "))
    abline(h=mean(res[,i]),col="green")
    lines(1:nDraw, cum_par[,i],col="red")
  }
  return(res)
}

mix_res1=mix_gib(nDraw = 2000)

## (c) Graphical comparison
yGrid <- seq(min(y)-sd(y),max(y)+sd(y),length = 1000)
yGridMin <- min(yGrid)
yGridMax <- max(yGrid)
# max(hist(y)$density)=0.01578841
ylim <- c(0,2*0.01578841)
hist(y,freq = FALSE,breaks = 20,
     xlim = c(yGridMin,yGridMax),
     ylim = ylim)
lines(yGrid,dnorm(yGrid, mean = colMeans(gib_sam)[1],
                  sd=sqrt(colMeans(gib_sam)[2])),col="blue")
## mean of each parameters in mixture model
meanPi=colMeans(mix_res2)[1]
meanMu1=colMeans(mix_res2)[2]
meanMu2=colMeans(mix_res2)[3]
meanSigma2_1=colMeans(mix_res2)[4]
meanSigma2_2=colMeans(mix_res2)[5]
mixDensity=meanPi*dnorm(yGrid,mean =meanMu1,sd=sqrt(meanSigma2_1))+
  (1- meanPi)*dnorm(yGrid,mean =meanMu2,sd=sqrt(meanSigma2_2))
lines(yGrid,mixDensity,col="red")

## mean of all draws densities of each value of yGrid
all_density=matrix(0,nrow = nrow(mix_res2),ncol = length(yGrid))
for (i in 1:nrow(mix_res2)) {
  all_density[i,]=mix_res2[i,1]*dnorm(yGrid,
  mean =mix_res2[i,2],sd=sqrt(mix_res2[i,4]))+
    (1- mix_res2[i,1])*dnorm(yGrid,mean=mix_res2[i,3],
                             sd=sqrt(mix_res2[i,5]))
}
fina_mean=colMeans(all_density)
lines(yGrid,fina_mean,col="green")


# 2. Metropolis Random Walk for Poinsson regression.
eBay=read.table("eBayNumberOfBidderData.dat", header = TRUE)

## (a) Use glm to obtain the MLE esitamtor of beta
glmModel=glm(nBids~0+., data = eBay, family = poisson)
summary(glmModel)
## (b) Approximate posterior density of beta
X=as.matrix(eBay[,2:ncol(eBay)])
y=eBay$nBids
nPara=ncol(X)
mu=rep(0,nPara)
Sigma=100*solve(t(X)%*%X)

LogPostPoisson <- function(beta,y,X,mu,Sigma){
  ### log likelihood
  logLik=0
  for (i in 1:length(y)) {
  logLik=logLik+y[i]*X[i,]%*%beta-exp(X[i,]%*%beta)-log(factorial(y[i]))
  }

  ### log prior
  logPrior <- dmvnorm(beta, mean = mu, Sigma, log=TRUE)

  # return log posterior
  return(logLik + logPrior)
}

LogPostPoisson(Mchain[1,],y,X,mu,Sigma)

init_beta=rep(0,nPara)
OptimResults=optim(init_beta,LogPostPoisson,
                   gr=NULL,y,X,mu,Sigma,method=c("BFGS"),
                   control=list(fnscale=-1),hessian=TRUE)

postMode=OptimResults$par
J_inver <- -solve(OptimResults$hessian)

## (c) Random Walk Metropolist algorithm for arbitrary posterior density

RWMSampler <- function(c,Jinverse,nDraw,logPostFunc,...){
  Mchain=matrix(0,nrow = nDraw,ncol = ncol(Jinverse))
  cum_par=matrix(0,nrow = nDraw,ncol = ncol(Jinverse))
  accp_num=0
  # intialize the theta_0, the first row of Mchain matrix
  Mchain[1,]=rep(1,ncol(Jinverse))
  cum_par[1,]=rep(1,ncol(Jinverse))
  for (i in 2:nDraw) {
```

```r
  # theta_p need to translate to a vector for the logPostPoisson
   theta_p=as.vector(rmvnorm(1,mean = Mchain[i-1,],sigma = c*Jinverse))
    alpha=min(1,exp(logPostFunc(theta_p,...)-logPostFunc(Mchain[i-1,],...)))
    u=runif(1)
    if(u<alpha){
      Mchain[i,]=theta_p
      accp_num=accp_num+1
      }
    else{Mchain[i,]=Mchain[i-1,]}
    cum_par[i,]=colMeans(Mchain[1:i,])
  }
  print(accp_num/nDraw)

  for (i in 1:9) {
    plot(1:nDraw,Mchain[,i],type = "l",
         xlab = "number of draw",ylab = paste("Beta_",i,sep = ""),
         main = paste(nDraw," draws of Beta_",i,sep = ""))
    abline(h=colMeans(Mchain)[i],col="green")
    lines(1:nDraw,cum_par[,i],col="red")
  }
  return(Mchain)
}

MC_res=RWMSampler(c=0.6,Jinverse = J_inver,nDraw = 10000, logPostFunc =LogPostPoisson,y,X,mu,Sigma)

## (d) Predictive distribution of the number of bidders
xNew=c(1,1,1,1,0,0,0,1,0.5)
predictive <- function(beta,newdata){
  coun_of_y=c(0:12)
  all_prob=rep(0,length(coun_of_y))
  pred_count=rep(0,nrow(beta))
  for (i in 1:nrow(beta)) {
    lamb=exp(newdata%*%beta[i,])
    all_prob=lamb^coun_of_y*exp(-lamb)/factorial(coun_of_y)
    pred_count[i]=which.max(all_prob)-1
  }
  return(pred_count)
}

samp_beta=MC_res[3001:10000,]
pred_y_2=predictive(beta = samp_beta,newdata = xNew)
barplot(table(pred_y_2))
for (i in 0:12) {
  dis_prob[i+1]=length(which(pred_y_2==i))
}

new3=data.frame()
string=c("0","1","2","3","4",
         "5","6","7","8","9","10","11","12")
new3=rbind(Num=string,cou=dis_prob,pro=dis_prob/nrow(samp_beta))
```