Bayesian Learning - lab3

Joris van Doorn , Weng Hang Wong

4/15/2020

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 1/100 inch, and records of zero end of 1983, of precipitation (rain or snow in units of 100 precipitation are excluded) at Snoqualmie Falls, Washington. Analyze the data using the following two models.

(a) Normal model.

Assume the daily precipitation $\{y_1, ..., y_n\}$ are independent normally distributed, $y_1, ..., y_n|\mu, \sigma^2 \sim N(\mu, \sigma^2)$ where both μ and σ^2 are unknown. Let $\mu \sim N(\mu_0, \tau_0^2)$ independently of $\sigma^2 \sim Inv - \chi^2(\nu_0, \sigma_0^2)$

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

The conditionally conjugate prior:

$$\mu \sim N(\mu_0, \tau_0^2)$$

$$\sigma^2 \sim Inv - \chi^2(v_0, \sigma_0^2)$$

The full conditional posteriors:

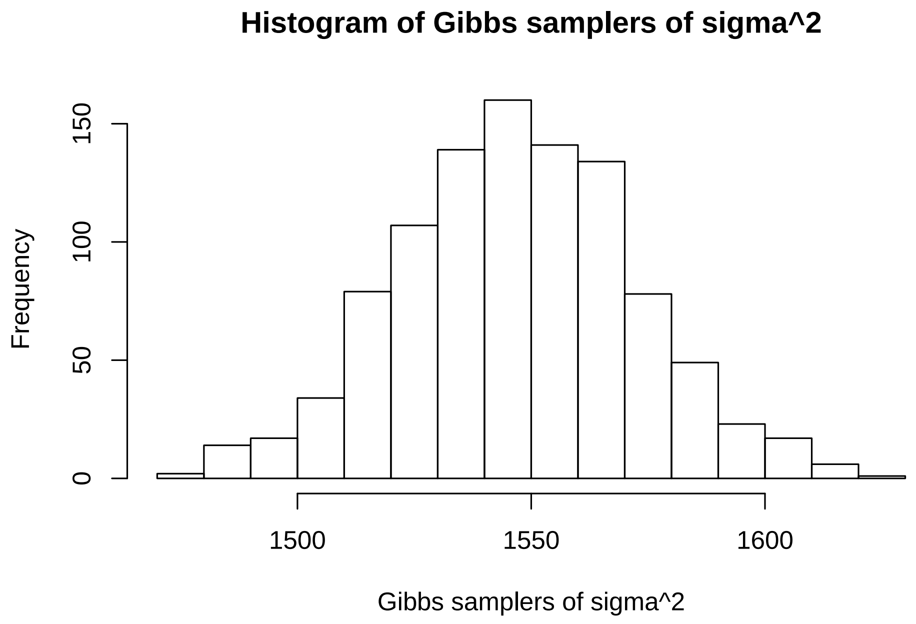
$$\mu|\sigma, x \sim N(\mu_n, \tau_n^2)$$
$$\sigma^2|\mu, x \sim Inv - \chi^2(v_n, \frac{v_0\sigma_0^2 + \sum_{i=1}^n (x_i - \mu)^2}{n + v_0})$$
$$where, \frac{1}{\tau_n^2} = \frac{n}{\sigma^2} + \frac{1}{\tau_0^2}$$
$$\mu_n = w\bar{x} + (1 - w)\mu_0$$

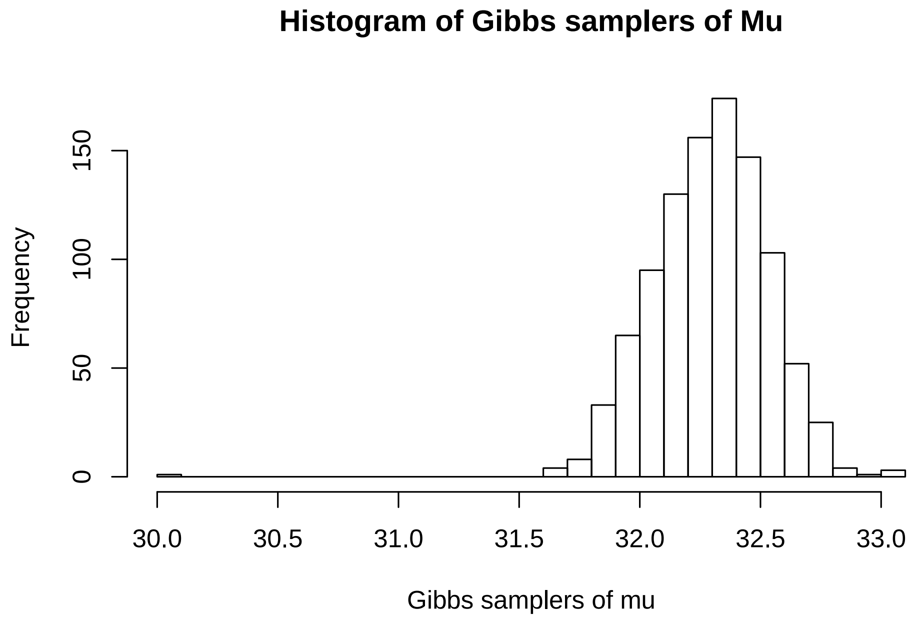
$$where, w = \frac{n/\sigma^2}{n/\sigma^2 + 1/\tau_0^2}$$

Since we don't have enough information about the percipitaion in Washington, base on lack of knowledge, we set our parameters as following: $\mu_0 = 30, \tau_0^2 = 50, \nu_0 = 3, \sigma_0^2 = var(givendata)$.

By using Gibbs Sampler, we simulate the mean and the variance from the joint posterior in a 1000 draws.

```
## [1] 1547.103
```



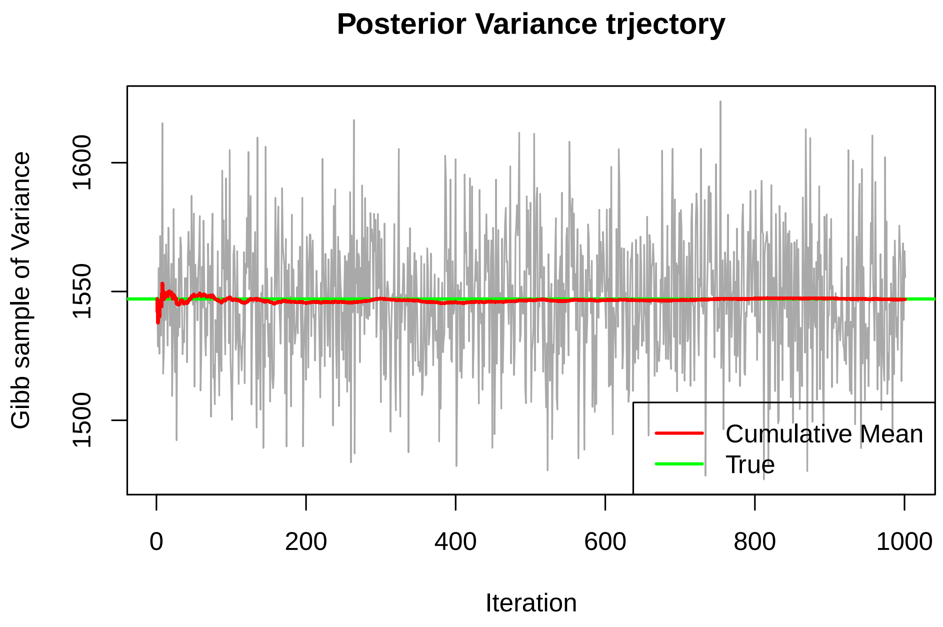


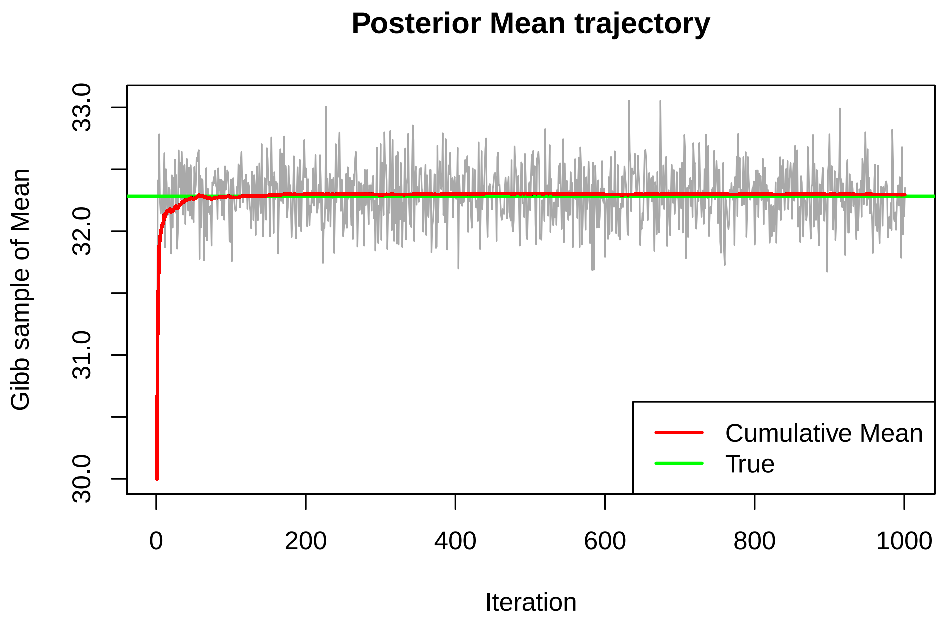
ii. Analyze the daily precipitation using your Gibbs sampler in (a)-i. Evaluate the convergence of the Gibbs sampler by suitable graphical methods, for example by plotting the trajectories of the sampled Markov chains.

The below two graphs are shown the trajectory plot of the sampled Markov chains. From the first graph of Posterior Variance trajectory, after around 50 iterations burning period, it is towards converaged to the true variance of the data which is about 1547.

From the second graph of Posterior Mean trajectory, the burning-period of which is faster than the varanice, which within around 50 iterations. After that, it is converaged to the true mean of the data which is about 32.28.

From the result, we can say that the Gibbs Sampling is a success, since both Gibbs mean and variance are converaged to the true mean and variance. Moreover, the burning period is fast and within 50 iterations, based on the initiative parameters are set close to the given data.





```
## The value of Gibbs sampling of Mu: 32.29442
## The value of Gibbs sampling of Sigma^2: 1546.928
```

(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:

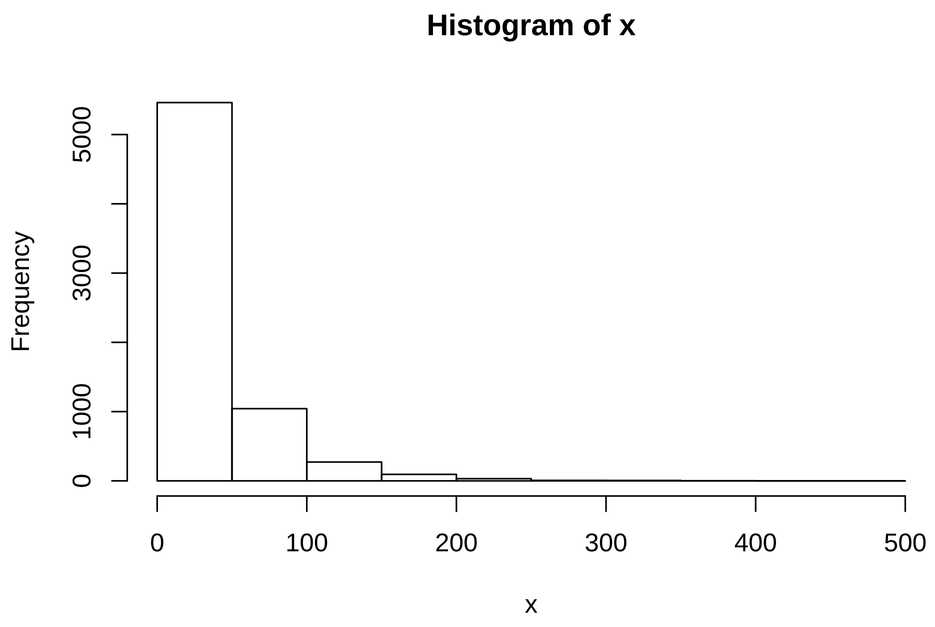
$$p(y_i|\mu, \sigma^2, \pi) = \pi N(y_i|\mu_1, \sigma_1^2) + (1-\pi) N(y_i|\mu_2, \sigma_2^2),$$

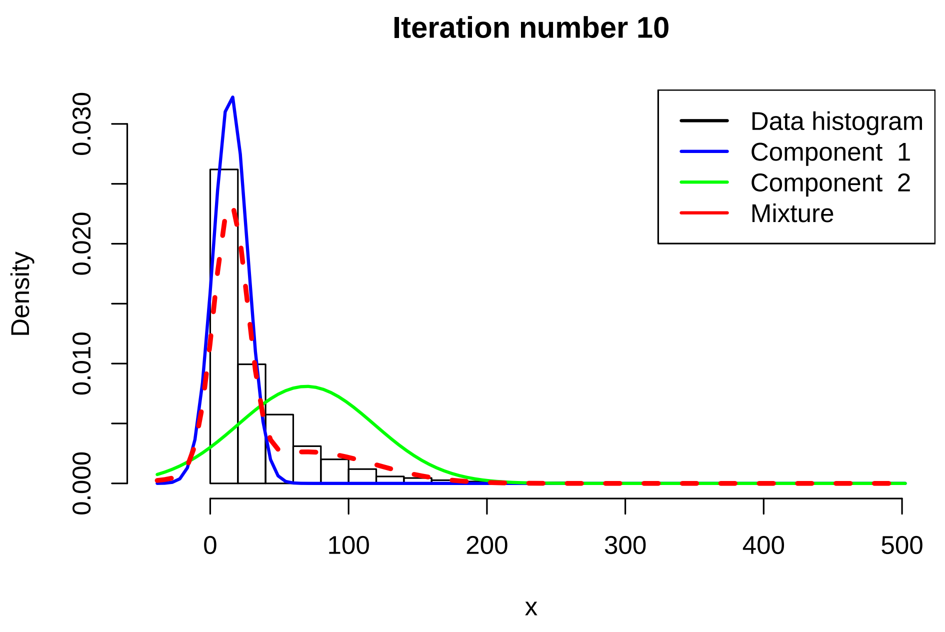
where

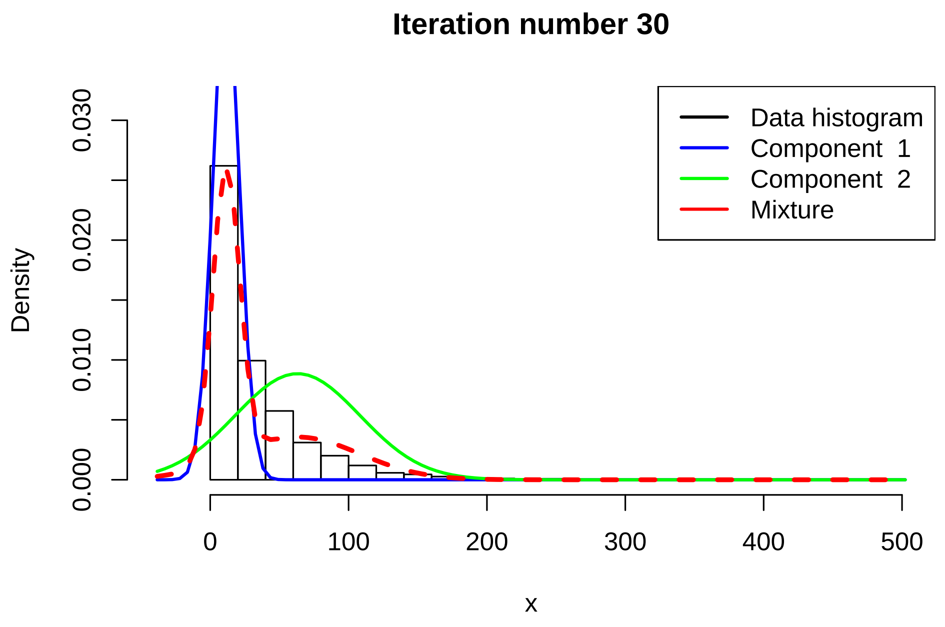
$$\mu = (\mu_1, \mu_2) \;\; and \;\; \sigma^2 = (\sigma_1^2, \sigma_2^2)$$

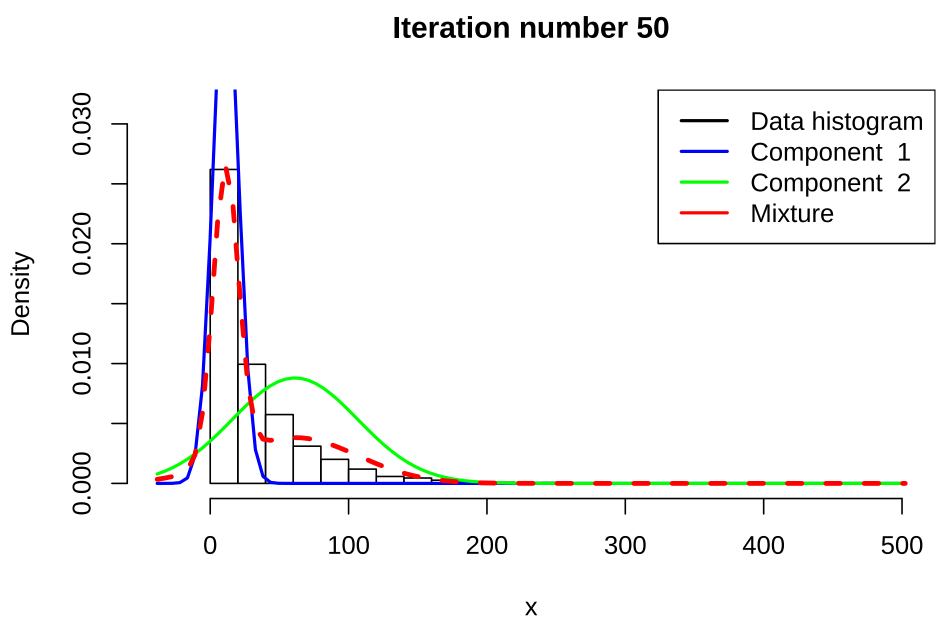
Use the Gibbs sampling data augmentation algorithm in NormalMixtureGibbs.R (available under Lecture 7 on the course page) to analyze the daily precipitation data. Set the prior hyperparameters suitably. Evaluate the convergence of the sampler.

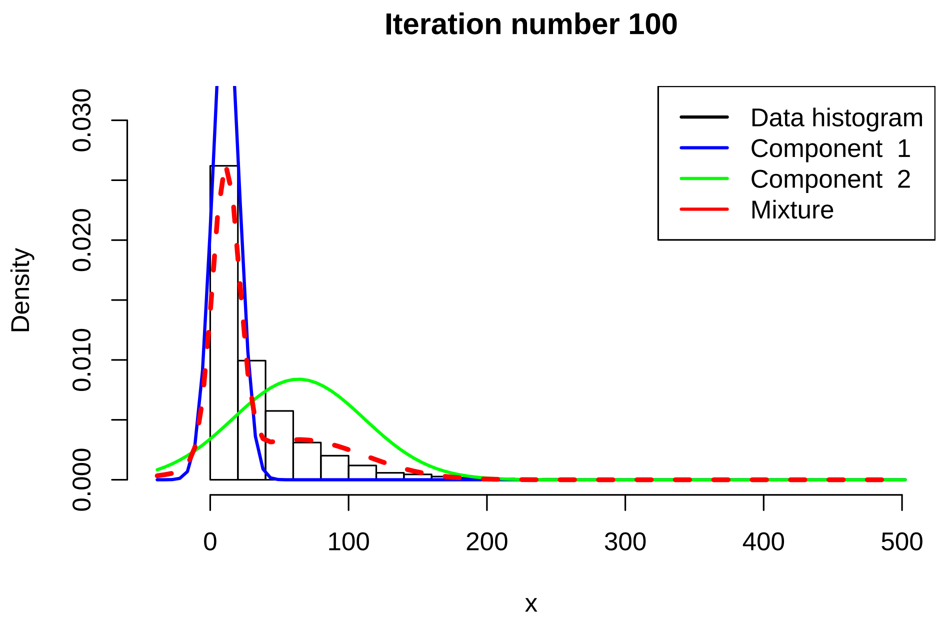
We set the same prior hyperparmeters as above. From 100 iterations, the sampler converaged very fast from the first 20 iterations, and after 30th iterations the components do not change much. Therefore, we can say in the algorithm of normal mixture, Gibbs sampling with the initial hyperparameters work well in fitting the data.




Iteration number 10
Density
0.000
0.010
0.020
0.030
0
100
200
300
400
500
x
Data histogram
Component 1
Component 2
Mixture


Iteration number 30
Density
0.000
0.010
0.020
0.030
0
100
200
300
400
500
x
Data histogram
Component 1
Component 2
Mixture



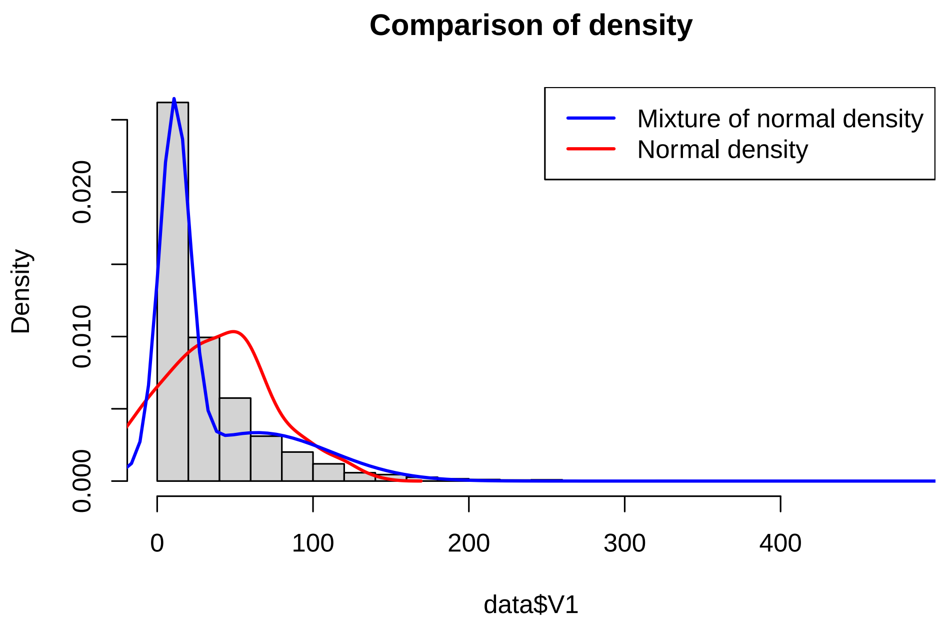


(c) Graphical comparison.

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.

Comparing with the Mixture of normal Density in (b) and Normal Density from (a), we can easily spot that the Normal Density from (a) does not fit well on the data density. However, using the mixture of normal

Density is fitted better than the nornal density one, though it is not perfect, but still can cover most of the density of the original data.



2. Metropolis Random Walk for Poisson regression.

Consider the following Poisson regression model

$$y_i|\beta \sim Poisson[exp(x_i^T\beta)], \quad i = 1, ..., n,$$

where y_i is the count for the ith observation in the sample and x_i 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?

```
## The estimator of beta are: 1.072442 -0.02054076 -0.3945165 0.4438426 -0.05219829 -0.2208712 0.070672
```

```
##
## 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
```

By using glm() to obtain the summary of the Posisson regression model, the most signficant covariates are having the most ***, which are VerifyID, Sealed, LogBook and MinBidShare. BajBlem obtain only one star so it is less significant. And the rest covariate are not influenced to our model in this case.

From the significant covariates we can tell that, the most influenced factor to the number of Bids is MinBid-Share, the lower the selling price is more appealing to bids.

(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:

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

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)

Likelihood of Poisson Regression

$$p(y|X, \tilde{\beta}) = \prod_{i=1}^{n} \frac{exp(y_i \tilde{\beta} x_i) - exp(-exp(\tilde{\beta x_i}))}{y_i!}$$

Log-likelihood of poisson regression

$$p(y_i|x_i,\tilde{\beta}) = \sum_{i=1}^{n}(y_i\tilde{\beta}x_i - exp(\tilde{\beta}x_i) - log(y_i!))$$

	Const	PowerSeller	VerifyID	Sealed	Minblem	MajBlem	LargNeg	LogBook	M
Const	0.0009455	-0.0007139	-0.0002742	-0.0002709	-0.0004455	-0.0002772	-0.0005128	0.0000644	
PowerSeller	-0.0007139	0.0013531	0.0000402	-0.0002949	0.0001143	-0.0002083	0.0002802	0.0001182	
VerifyID	-0.0002742	0.0000402	0.0085154	-0.0007825	-0.0001014	0.0002283	0.0003314	-0.0003192	
Sealed	-0.0002709	-0.0002949	-0.0007825	0.0025578	0.0003577	0.0004532	0.0003376	-0.0001311	
Minblem	-0.0004455	0.0001143	-0.0001014	0.0003577	0.0036246	0.0003492	0.0000584	0.0000585	
MajBlem	-0.0002772	-0.0002083	0.0002283	0.0004532	0.0003492	0.0083651	0.0004049	-0.0000898	
LargNeg	-0.0005128	0.0002802	0.0003314	0.0003376	0.0000584	0.0004049	0.0031751	-0.0002542	
LogBook	0.0000644	0.0001182	-0.0003192	-0.0001311	0.0000585	-0.0000898	-0.0002542	0.0008385	
MinBidShare	0.0011099	-0.0005686	-0.0004293	-0.0000576	-0.0000644	0.0002622	-0.0001063	0.0010374	

	Approximate Beta tilde
Const	1.0698412
PowerSeller	-0.0205125
VerifyID	-0.3930060
Sealed	0.4435555
Minblem	-0.0524663
MajBlem	-0.2212384
LargNeg	0.0706968
LogBook	-0.1202177
MinBidShare	-1.8919850

Comparing to the maximum likelihood estimator β we obtained in (a), the posterior mode $\tilde{\beta}$ the result obtained by numerical optimization is very close to β.

(c)

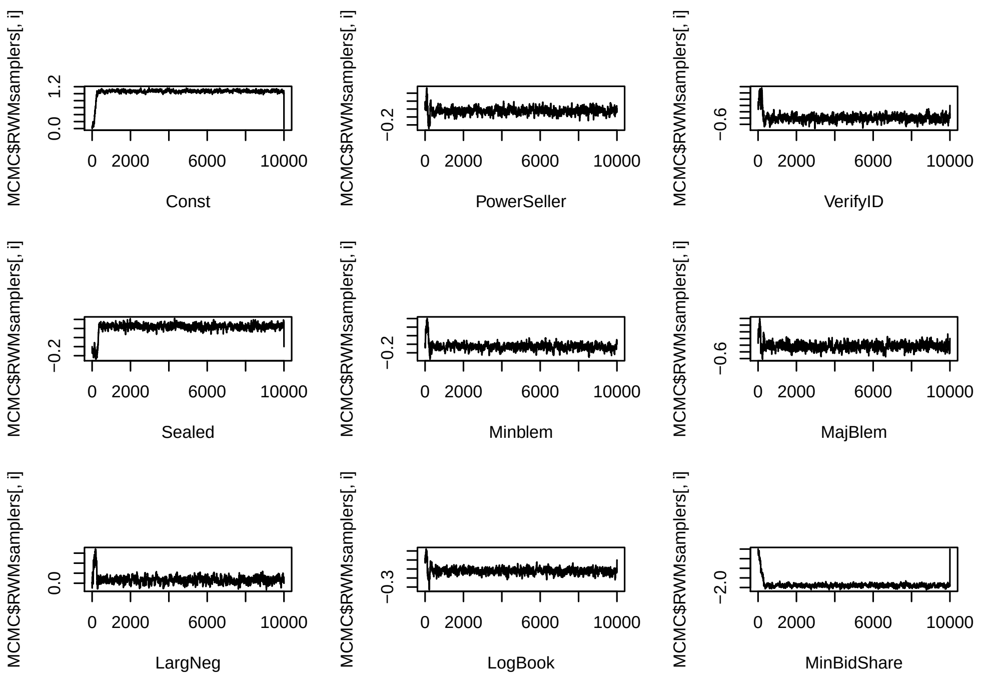
Now, let's simulate from the actual posterior of β using the Metropolis algorithm and compare with the approximate results in b). Program a general function that uses the Metropolis algorithm to generate random draws from an arbitrary posterior density. In order to show that it is a general function 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):

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

where $\sum = J_y^{-1}(\hat{\beta})$ obtained in b). The value c is a tuning parameter and should be an input to your Metropolis function. The user of your Metropolis 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.

In order to sample from the proposal density, we use Ramdom Walk metropolis sampler with the samples from the posterior of β from(b). In the RWM function, we use the constant c=0.6, in order to get the average acceptance probability within 25-30%. After the sampling, we calculated the average acceptance is 28% here.

Take a look on the graph which contains the 9 sampling plots of 9 different covariates of the data. Each graph is converaged in 10000 iterations. The following table also indicates the mean of the sampling Betas, which are very close to the sampling from (b).



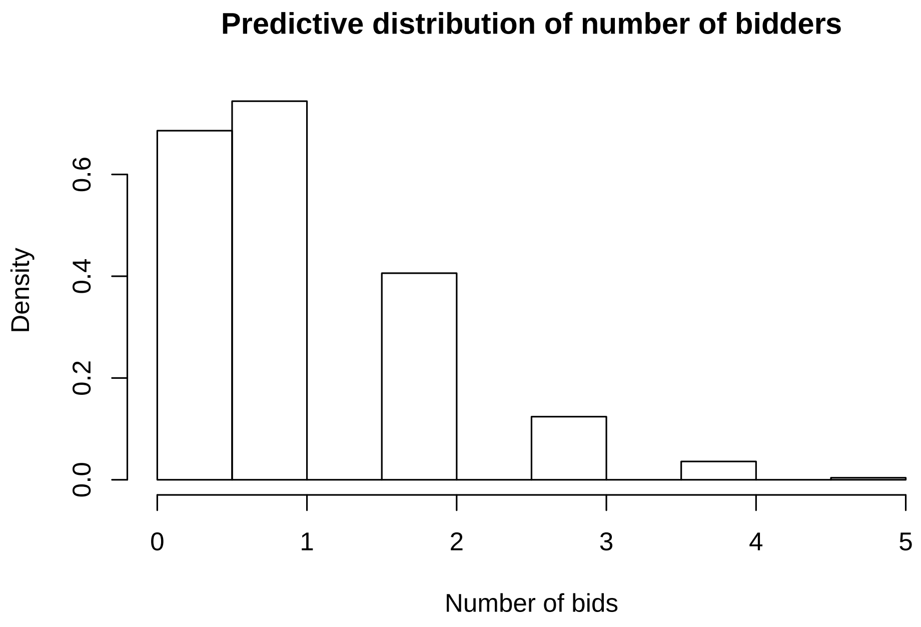
```
## The average acceptance probability: 0.2808045
```

	Beta Means
Const	1.0548221
PowerSeller	-0.0222561
VerifyID	-0.3875125
Sealed	0.4292535
Minblem	-0.0520769
MajBlem	-0.2214747
LargNeg	0.0715963
LogBook	-0.1212416
MinBidShare	-1.8664133

(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

From the below graph, we can the simulation of the predictive distubuiton of number of bidders. And the probability of no bidders in this new auction is 34.3%.



```
## The probability of no bidders in this new auction: 0.343
```

Appendix

```
library(LaplacesDemon)
# 1.
## (a)
###.i. Gibbs sampler from joint posterior
data = read.table("rainfall.dat")

# para from data
n = length(row(data)) #6920
xbar = mean(data$V1) #32.28
var(data$V1)

#set up parameters
mu0 = 30 #True mean =32.28
tau20 = 150 # we dont have prior knowledge
nu0 = 3 #df set it to a small value
sigma20 = var(data$V1) #True var=1547.103

## Gibbs sampling
Ndraws = 1000
GibbMu = c()
GibbMu[1] = mu0
```

```r
GibbSigma2 = c()
GibbSigma2[1] = sigma20

set.seed(12345)
for(i in 1:Ndraws){
  # parameters w mu tau, update mu_n, tau_n
  w = (n/GibbSigma2[i]) / ( (n/GibbSigma2) + (1/tau20) )
  mu_n = w*xbar + (1-w)*mu0
  tau_n = 1/ ( (n/GibbSigma2[i]) + (1/tau20)  )

  # sampling posterior mu, add in mu
  GibbMu[i+1] = rnorm(1, mean=mu_n, sd=tau_n)

  #sampling posterior sigma2, add in sigma2
  nu_n = nu0 + n
  Scale =(nu0*sigma20 + sum((data$V1 - GibbMu[i+1])^2) )/ (n+nu0)
  GibbSigma2[i+1] = rinvchisq(1, df= nu_n, scale= Scale)
}

hist(GibbSigma2,breaks=20, main="Histogram of Gibbs samplers of sigma^2", xlab="Gibbs samplers of sigma
hist(GibbMu, breaks=30, main="Histogram of Gibbs samplers of Mu", xlab="Gibbs samplers of mu")


### ii.
cumMu=c()
cumSigma2=c()

# plot variance MC converage
plot(GibbSigma2, type="l",col="darkgrey", main="Posterior Variance trjectory", xlab="Iteration",ylab="G
abline(h=var(data$V1), col="green", lwd=2)
legend("bottomright",legend=c("Cumulative Mean", "True"), col=c("red","green"),lwd=2)
for(i in 1:length(GibbSigma2)){
  cumSigma2[i] = mean(GibbSigma2[1:i])
  lines(cumSigma2, col="red", lwd=2)
}

# plot mean MC converage
plot(GibbMu, type="l", col="darkgrey", main="Posterior Mean trajectory", xlab="Iteration", ylab="Gibb s
abline(h=xbar, col="green", lwd=2)
legend("bottomright",legend=c("Cumulative Mean", "True"), col=c("red","green"),lwd=2)
for(i in 1:length(GibbMu)){
  cumMu[i]= mean(GibbMu[1:i])
  lines(cumMu,col="red", lwd=2)
}
cat("The value of Gibbs sampling of Mu:",mean(GibbMu))
cat("The value of Gibbs sampling of Sigma^2:", mean(GibbSigma2))


# 1.b

# Estimating a simple mixture of normals
# Author: Mattias Villani, IDA, Linkoping University. http://mattiasvillani.com
```

```
##########    BEGIN USER INPUT #################
# Data options
#data(faithful)
#rawData <- faithful
x <- as.matrix(data$V1)

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

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(30,nComp) # Prior mean of mu
tau2Prior <- rep(150,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

# MCMC options
nIter <- 100 # 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(x)
```

```r
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))


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))
  }

  # 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 
  }

  # 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(k %in% c(10,30,50,100) ){
  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("topright", box.lty = 1, legend = c("Data histogram",components, 'Mixture'), col = c("black"
    #Sys.sleep(sleepTime)
  }

  }
}

#########################    Helper functions    ##############################################




#1.c

##1. histogram of data
hist(data$V1,freq=FALSE, breaks=30, col="lightgrey",main="Comparison of density")

##2. normal density N(yi|mu,sigma2)
set.seed(12345)
NormalDens = rnorm(500, mean=mean(GibbMu), sd=sqrt(mean(GibbSigma2)))
lines(density(NormalDens), col="red",lwd=2)

##3. mixture normal density p(yi|mu, sigma2,pi)

lines(xGrid,mixDens, col="blue", lwd=2)
legend("topright", legend=c("Mixture of normal density", "Normal density"), col=c("blue","red"),lwd=2)

#2.a
data = read.table("eBayNumberOfBidderData.dat",header=T)

# obtain beta
data1 = data[,-2]
glm_model= glm(nBids~., family="poisson",data=data1)
cat("The estimator of beta are:",glm_model$coefficients)
summary(glm_model)


library(mvtnorm)

#2.b
y=data[,1]
x=as.matrix(data[,2:10]) #include constant
colnames(x) = names(data)[2:10]
```

```
nPara = dim(x)[2]

# Prior hyperpara
mu <- as.vector(rep(0,nPara))
sigma <- 100*solve(t(x)%*%x)

# Modified Mattias' code from lab2
# Log-posterior function = log(prior)+log(llh)
set.seed(12345)
LogPost = function(betaVec, y, x, mu, sigma){
  pred = x%*%betaVec
  #log LLH
  logLLH= sum(y*pred - exp(pred)-log(factorial(y)) )
  #log prior using dmvnorm from beta vector
  logPrior = dmvnorm(betaVec, mean=rep(0,length(betaVec)), sigma, log=T)
  res = logLLH + logPrior
  return(res)
}

initValue <- as.vector(rep(0,dim(x)[2]));
OptimResults<-optim(initValue,LogPost,gr=NULL,y,x,mu,sigma,method=c("BFGS"),control=list(fnscale=-1),hes

## find the value of Beta and Hessian
Beta_tilde = as.matrix(OptimResults$par)
hessianBeta = -solve(OptimResults$hessian) #we want -Inv Hessian
approx_PostStd <- as.matrix(sqrt(diag(hessianBeta)))

# Beta table
library(knitr)

#Covariance Matrix
colnames(hessianBeta) = names(data)[2:10]
rownames(hessianBeta) =  names(data)[2:10]
kable(hessianBeta)

# Beta tilde table
rownames(Beta_tilde) = names(data)[2:10]
colnames(Beta_tilde) = "Approximate Beta tilde"
kable(Beta_tilde)



library(mvtnorm)
#2.c

## random walk Metropolis
set.seed(12345)

RWMsampler= function(theta0, Ndraws, c, proposeSigma, LogPost, ...){

  #set gird for sampling theta
  theta = matrix(0, nrow=Ndraws+1, ncol=length(theta0))
  theta[1,] = theta0
```

```r
  alphas=c()

  i=2
  while(i < Ndraws){
    #simulate theta_p from sample proposal N(theta_i-1, c*Sigma)
    theta_p = as.vector(rmvnorm(1, mean=theta[i-1,], sigma= c*proposeSigma))

    #simulat U from runif
    U = runif(1, 0,1)

    #calculate alpha prob
    alpha = min(1, exp(LogPost(theta_p,...)-LogPost(theta[i-1,],...))) #use exp() to cancel log

    if(U<alpha){
      theta[i,]=theta_p
    }else{
      theta[i,]=theta[i-1,]
    }

    alphas[i-1]=alpha
    i=i+1
  }
  return(list(RWMsamplers=theta, alphas=alphas))
}

theta0=initValue
Ndraws=10000
c=0.6
proposeSigma=hessianBeta
MCMC = RWMsampler(theta0, Ndraws, c, proposeSigma, LogPost, y, x, mu, sigma)


# plot the 9 samples covaraite from MCMC
par(mfrow=c(3,3))
for(i in 1:9){
  plot(MCMC$RWMsamplers[,i],type="l",xlab=names(data)[i+1])
}

cat("The average acceptance probability:",mean(MCMC$alphas))

beta_means =colMeans(MCMC$RWMsamplers)
names(beta_means) = names(data)[2:10]
kable(beta_means,col.names = "Beta Means")

x_given =c(Const = 1,
PowerSeller = 1,
VerifyID = 1,
Sealed = 1,
MinBlem = 0,
MajBlem = 0,
LargNeg = 0,
LogBook = 1,
MinBidShare = 0.5)
```

```r
x_given=matrix(x_given,nrow=1, ncol=9)

set.seed(12345)
# simulate the prediction dist from possion[exp(t(x)*beta)]
pred_bids=c()
beta_means=matrix(beta_means, ncol=1)

for(i in 1:dim(data)[1]){
  pred_bids[i] = rpois(1,exp(x_given %*% beta_means))
}

hist(pred_bids,freq = F,main="Predictive distribution of number of bidders",xlab="Number of bids")

#the no bid prob
prob_0bids=length(which(pred_bids == 0))/dim(data)[1]

cat("The probability of no bidders in this new auction:", prob_0bids)
```