

lab3

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

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

The conditionally conjugate prior:

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

$$\sigma^2 \sim \text{Inv} - \chi^2(\nu_0, \sigma_0^2)$$

The full conditional posteriors:

$$\begin{aligned} \mu | \sigma, x &\sim N(\mu_n, \tau_n^2) \\ \sigma^2 | \mu, x &\sim \text{Inv} - \chi^2\left(\nu_n, \frac{\nu_0 \sigma_0^2 + \sum_{i=1}^n (x_i - \mu)^2}{n + \nu_0}\right) \end{aligned}$$

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

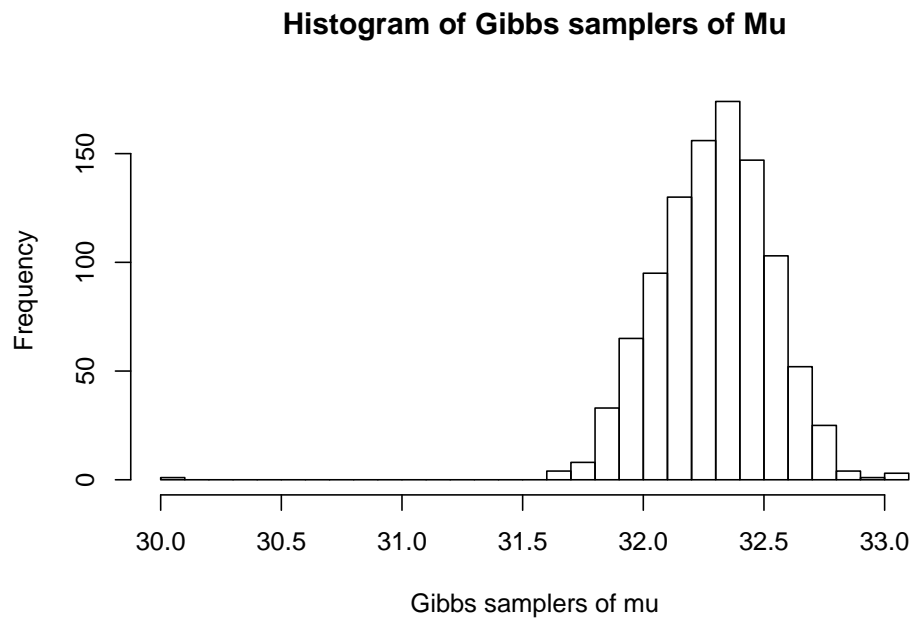
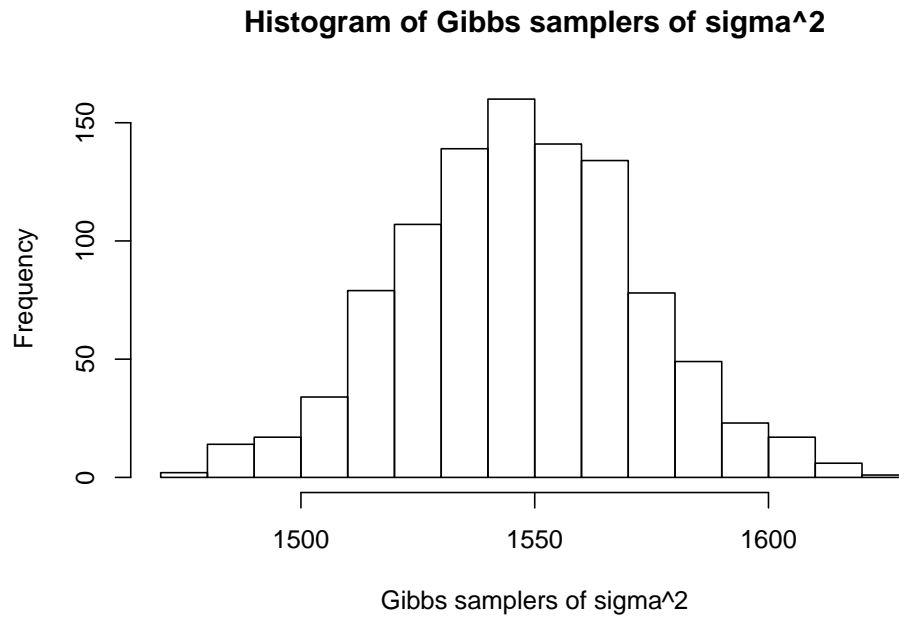
$$\mu_n = w\bar{x} + (1 - w)\mu_0$$

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

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

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

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



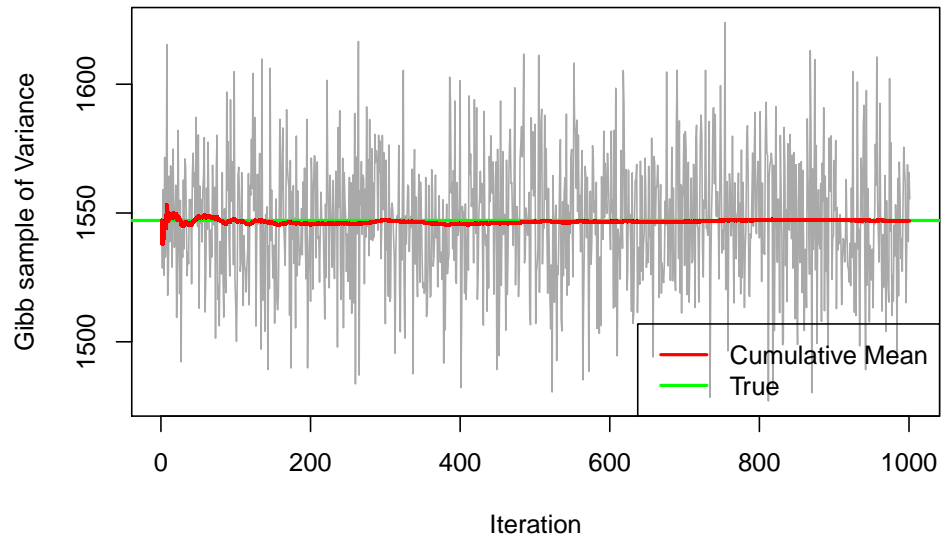
ii. Analyze the daily precipitation using your Gibbs sampler in (a)-i. Evaluate the convergence of the Gibbs sampler by suitable graphical methods, for example by plotting the trajectories of the sampled Markov chains.

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 converged to the true variance of the data which is about 1547.

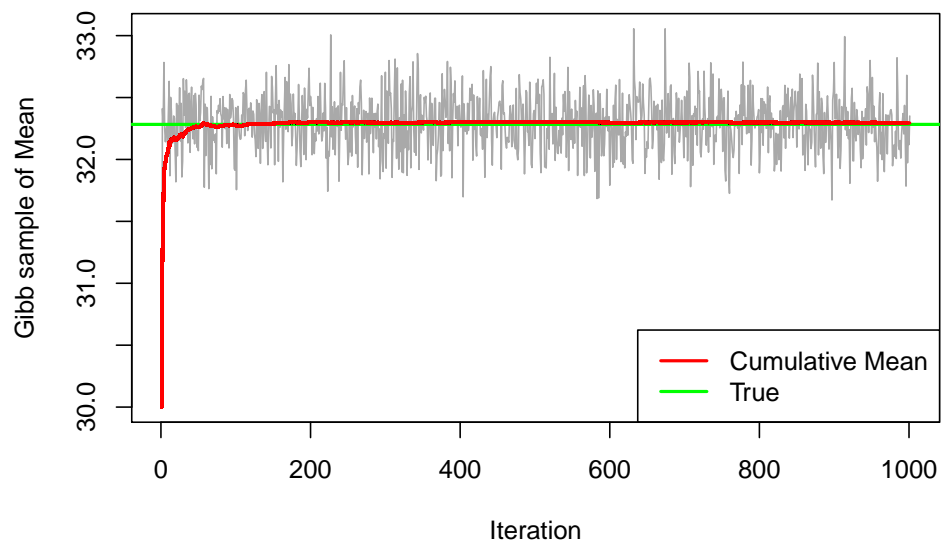
From the second graph of Posterior Mean trajectory, the burning-period of which is faster than the variance, which within around 50 iterations. After that, it is converged to the true mean of the data which is about 32.28.

From the result, we can say that the Gibbs Sampling is a success, since both Gibbs mean and variance are converged 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.

Posterior Variance trajectory



Posterior Mean trajectory



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, \dots, y_n follow an iid two-component mixture of normals model:

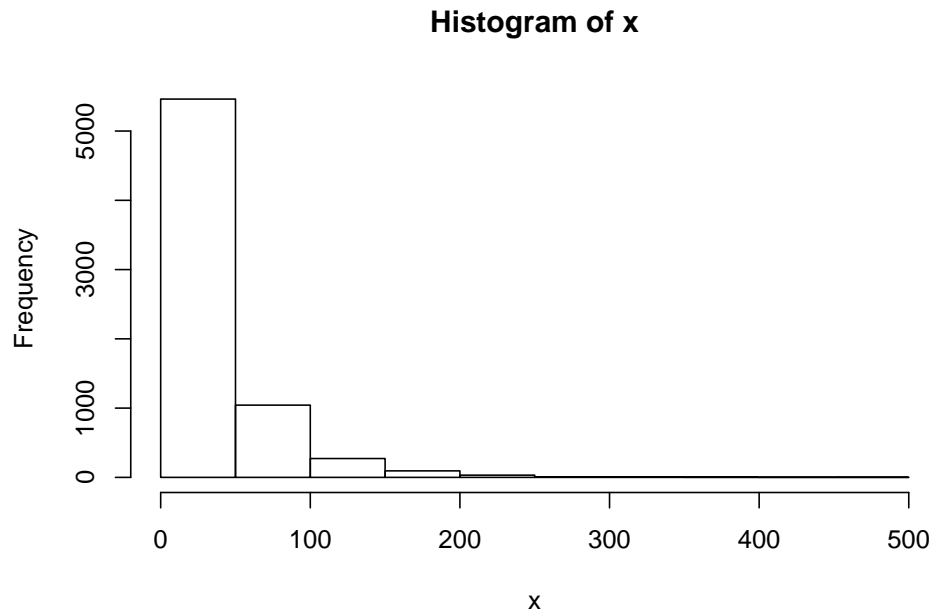
$$p(y_i|\mu, \sigma^2, \pi) = \pi N(y_i|\mu_1, \sigma_1^2) + (1 - \pi)N(y_i|\mu_2, \sigma_2^2),$$

where

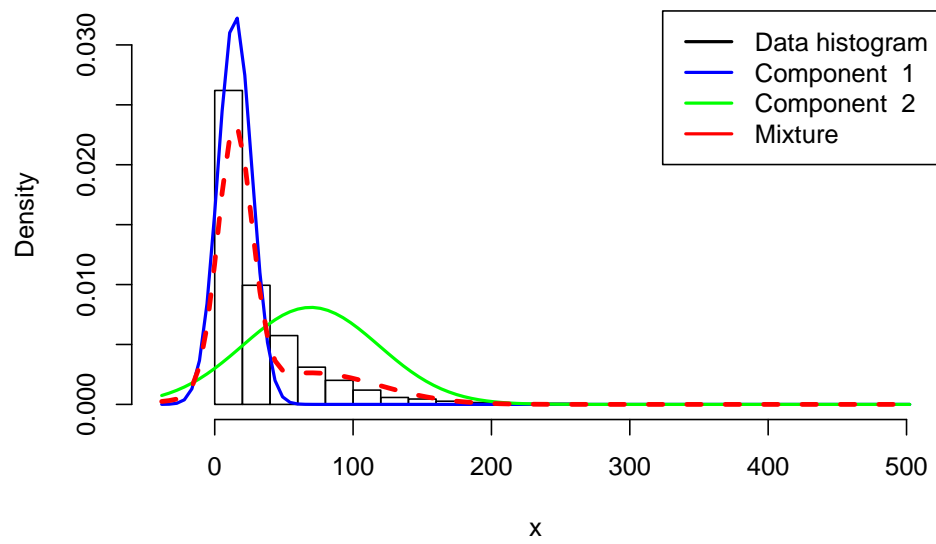
$$\mu = (\mu_1, \mu_2) \text{ and } \sigma^2 = (\sigma_1^2, \sigma_2^2)$$

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.

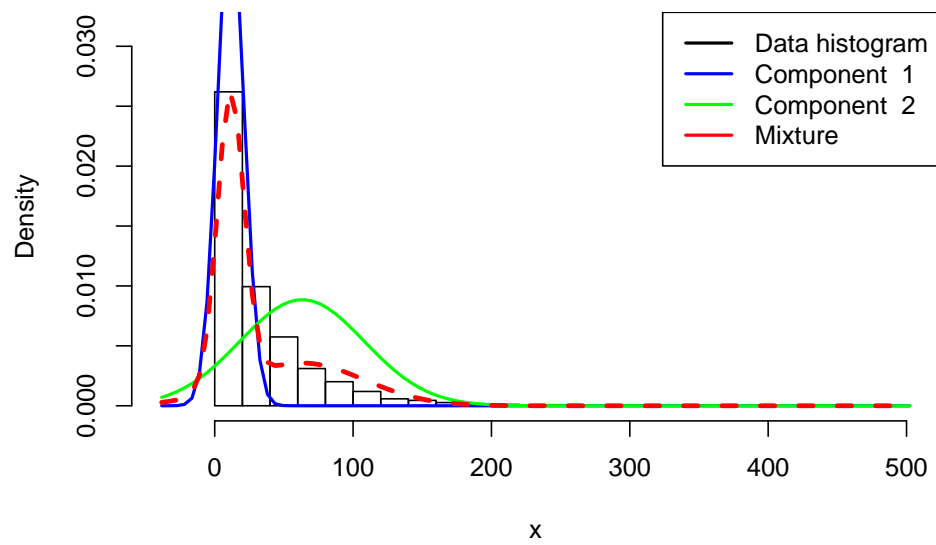
We set the same prior hyperparameters as above. From 100 iterations, the sampler converged very fast from the first 20 iterations, and after 30th iterations the components do not change much. Therefore, we can say in the algorithm of normal mixture, Gibbs sampling with the initial hyperparameters work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

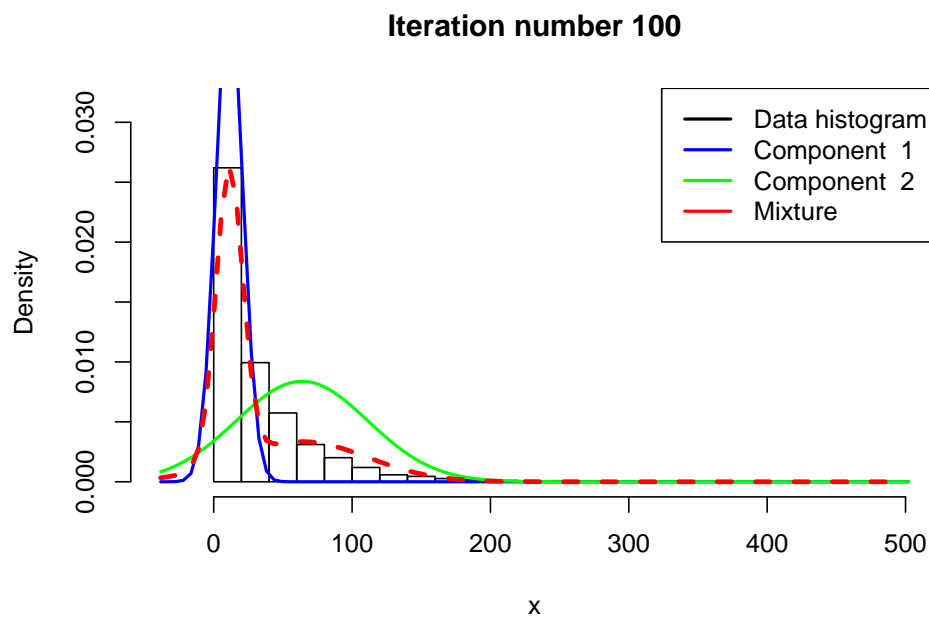
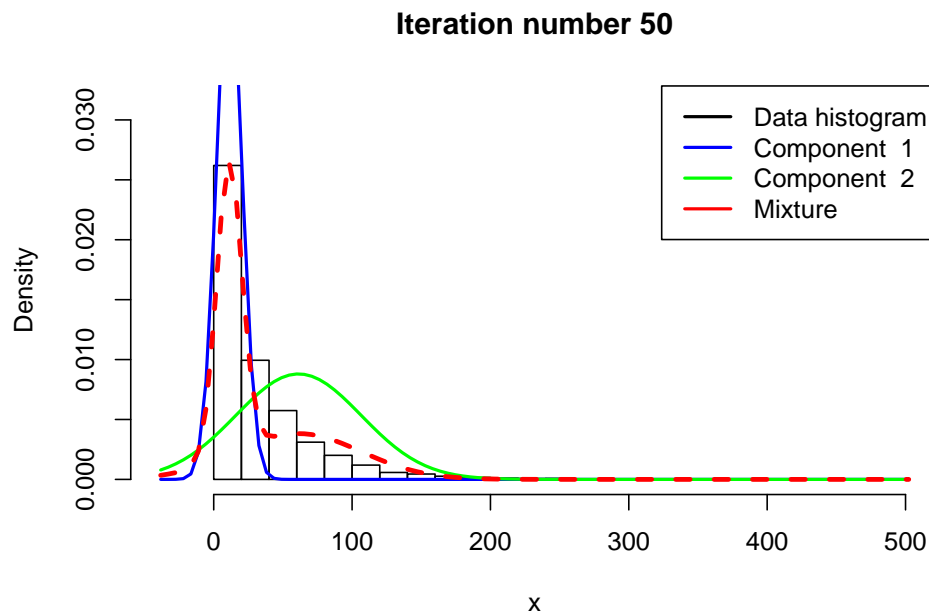


Iteration number 10



Iteration number 30



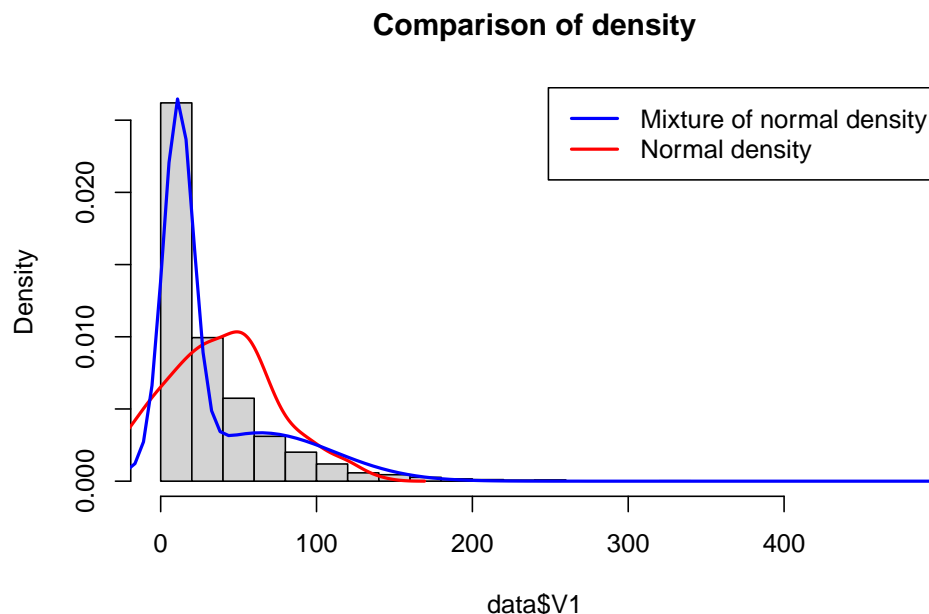


(c) Graphical comparison.

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 normal 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

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

where y_i is the count for the i th observation in the sample and x_i is the p -dimensional vector with covariate observations for the i th 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 Poisson regression model, the most significant covariates are having the most ***, which are `VerifyID`, `Sealed`, `LogBook` and `MinBidShare`, `BajBlem` obtain only one star so it is less significant. But 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 `MinBidShare`, 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 $n \times p$ covariate matrix. This is a commonly used prior which is called Zellner's g-prior. Assume first that the posterior density is approximately multivariate normal:

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

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

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
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] = rinvcchisq(1, df= nu_n, scale= Scale)
}

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

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

# plot variance MC convergence
plot(GibbSigma2, type="l",col="darkgrey", main="Posterior Variance trjectory", xlab="Iteration",ylab="Gibbs samplers of sigma^2")
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, Linköping 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) # Dividing every column of piDraws by the sum of the elements in that column
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)
S <- t(rmultinom(nObs, size = 1 , prob = rep(1/nComp,nComp))) # nObs-by-nComp matrix with component allocation
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 allocation
  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  $N(y_i|\mu,\sigma^2)$ 
set.seed(12345)
NormalDens = rnorm(500, mean=mean(GibbMu), sd=sqrt(mean(GibbSigma2)))
lines(density(NormalDens), col="red",lwd=2)

##3. mixture normal density  $p(y_i|\mu, \sigma^2,\pi)$ 

```

```

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)

```