732A91 - Lab 3

Joris van Doorn || Weng Hang Wong

17 May 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: $\mu_0 = 30, \tau_0^2 = 50, \nu_0 = 3, \sigma_0^2 = var(givendata)$.

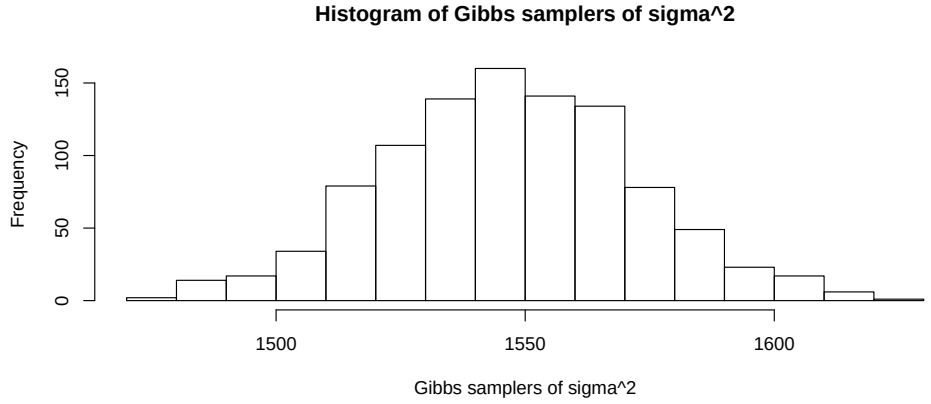
By using Gibbs Sampler, we simulate the mean and the variance from the joint posterior in a 1000 draws.

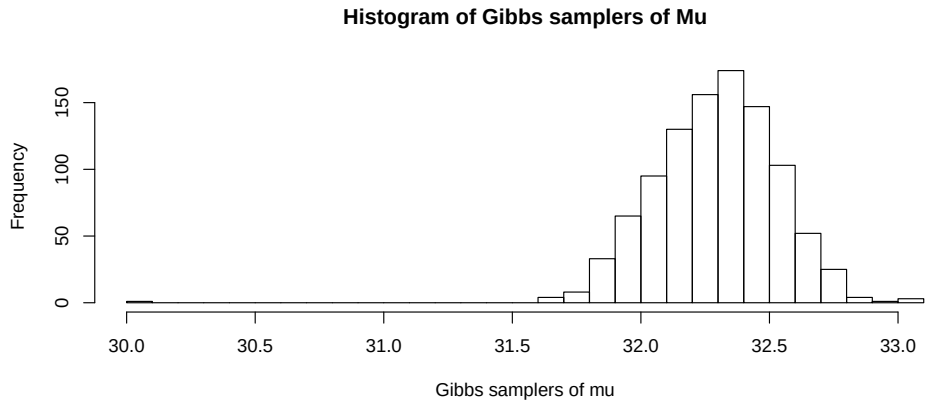
```
##
## Attaching package: 'LaplacesDemon'
```

```
## The following objects are masked from 'package:mvtnorm':
##
##     dmvt, rmvt

## The following object is masked from 'package:purrr':
##
##     partial

## [1] 1547.103
```

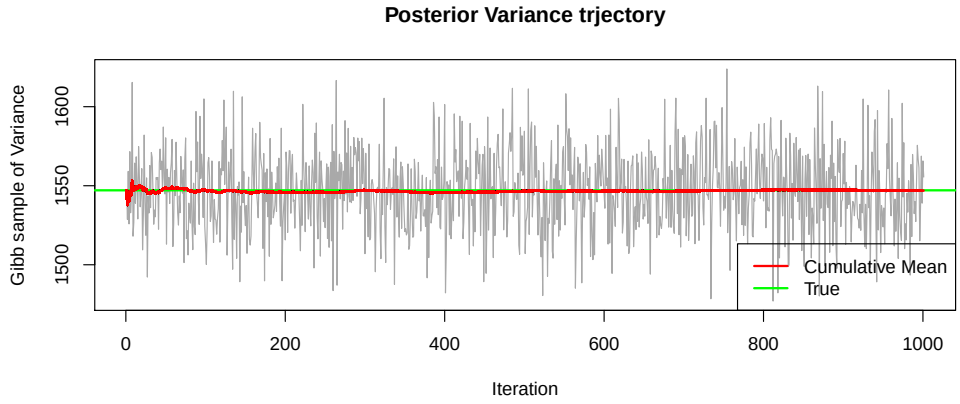


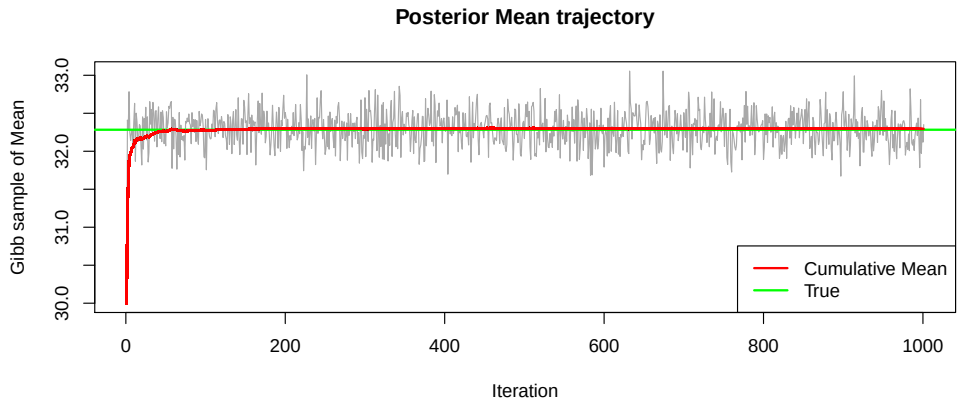


ii. Analyze the daily precipitation using your Gibbs sampler in (a)-i. Evaluate the convergence of the Gibbs sampler by suitable graphical methods, for example by plotting the trajectories of the sampled Markov chains. The below two graphs are shown the trajectory plot of the sampled Markov chains. From 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:

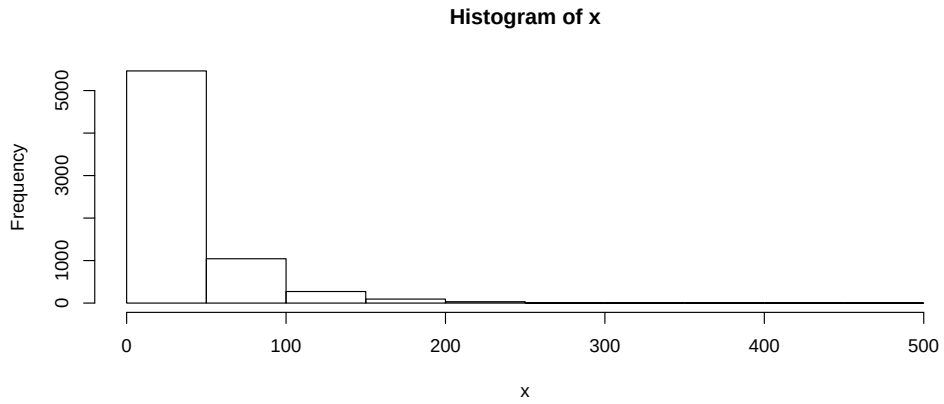
$$p(y_i|\mu, \sigma^2, \pi) = \pi N(y_i|\mu_1, \sigma_1^2) + (1 - \pi)N(y_i|\mu_2, \sigma_2^2),$$

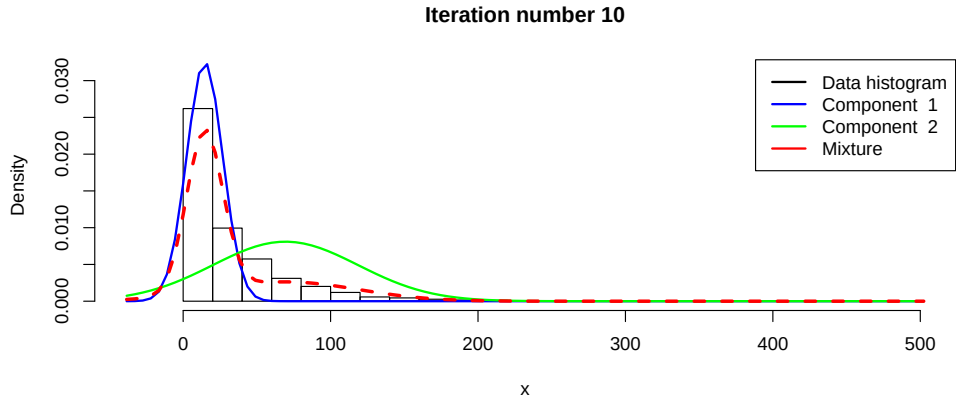
where

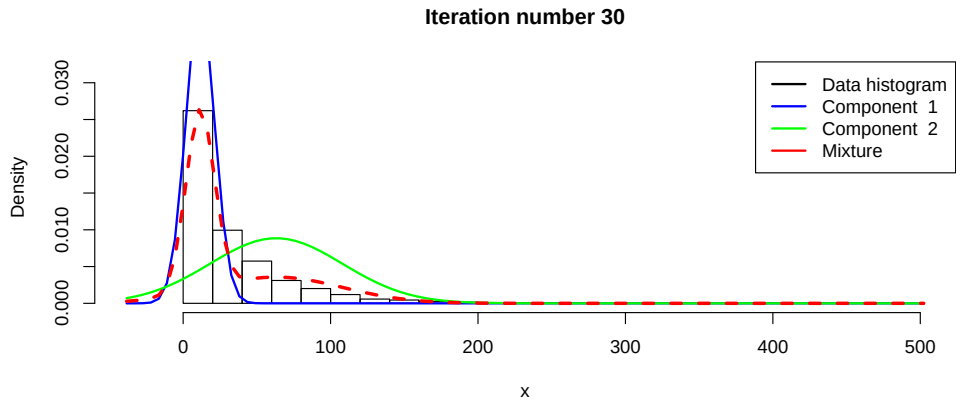
$$\mu = (\mu_1, \mu_2) \quad and \quad \sigma^2 = (\sigma_1^2, \sigma_2^2)$$

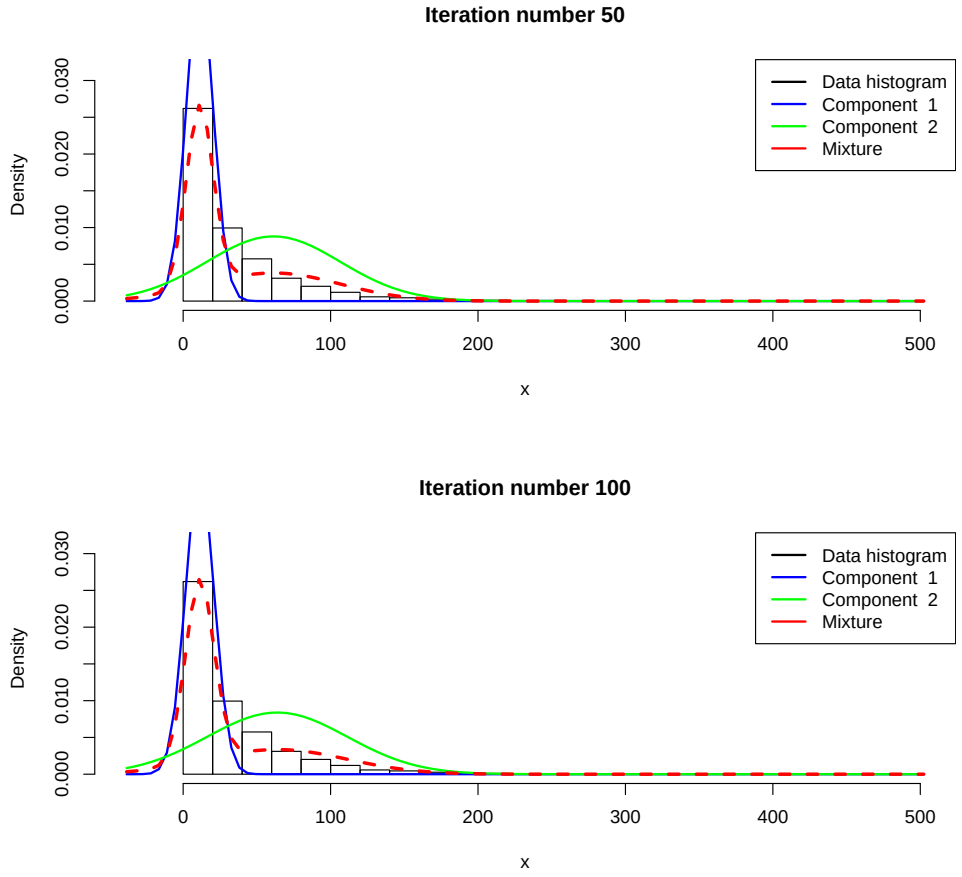
Use the Gibbs sampling data augmentation algorithm in NormalMixtureGibbs.R (available under Lecture 7 on the course page) to analyze the daily precipitation data. Set the prior hyperparameters suitably. Evaluate the convergence of the sampler.

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.


Histogram of x
Frequency
0
1000
3000
5000
0
100
200
300
400
500
x


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

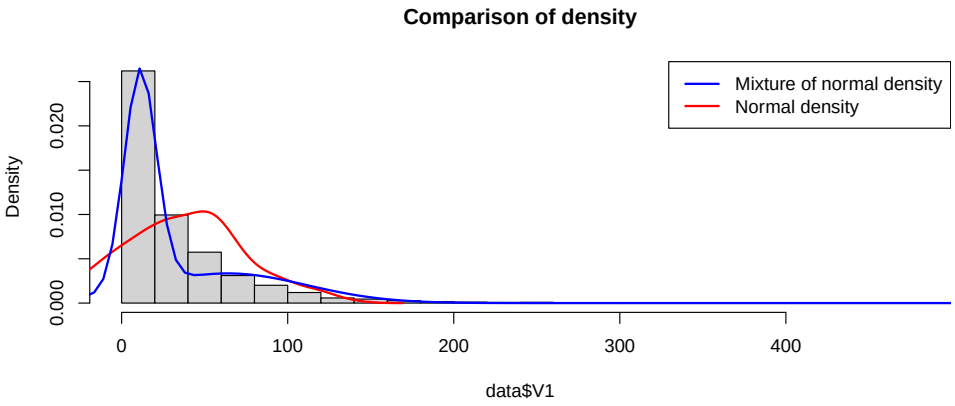

Iteration number 30
Density
0.000
0.010
0.020
0.030
0
100
200
300
400
500
x
Data histogram
Component 1
Component 2
Mixture



(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.


Comparison of density
Mixture of normal density
Normal density
Density
0.000
0.010
0.020
0
100
200
300
400
data$V1

2. Metropolis Random Walk for Poisson regression.

Consider the following Poisson regression model

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

where yi 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 opned 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?

```
##
## Call:
## glm(formula = Y ~ X, family = poisson(link = "log"))
##
## 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 ***
## XPowerSeller -0.02054    0.03678  -0.558   0.5765
## XVerifyID    -0.39452    0.09243  -4.268 1.97e-05 ***
## XSealed       0.44384    0.05056   8.778  < 2e-16 ***
## XMinblem     -0.05220    0.06020  -0.867   0.3859
## XMajBlem     -0.22087    0.09144  -2.416   0.0157 *
## XLargNeg      0.07067    0.05633   1.255   0.2096
## XLogBook     -0.12068    0.02896  -4.166 3.09e-05 ***
## XMinBidShare -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
```

The intercept, VerifyID, Sealed, Logbook, and MinBidShare are al significant with $p < 0.0001$. Furthermore is MajBlem significant at $p < 0.01$. PowerSeller, Minblem, and LargNeg do not appear to be significant.

b.

Let's now do a Bayesian analysis of the Poisson regression. Let the prior be $\beta \sim N[0, 100(X^T X)^{-1}$ *where* X *is the nxp 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.).*

	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	

	Verification	Beta_hat	Beta_std
(Intercept)	1.0724421	1.0698412	0.0307484
XPowerSeller	-0.0205408	-0.0205125	0.0367842
XVerifyID	-0.3945165	-0.3930060	0.0922787
XSealed	0.4438426	0.4435555	0.0505745
XMinblem	-0.0521983	-0.0524663	0.0602047
XMajBlem	-0.2208712	-0.2212384	0.0914607
XLargNeg	0.0706725	0.0706968	0.0563477
XLogBook	-0.1206776	-0.1202177	0.0289564
XMinBidShare	-1.8940966	-1.8919850	0.0710968

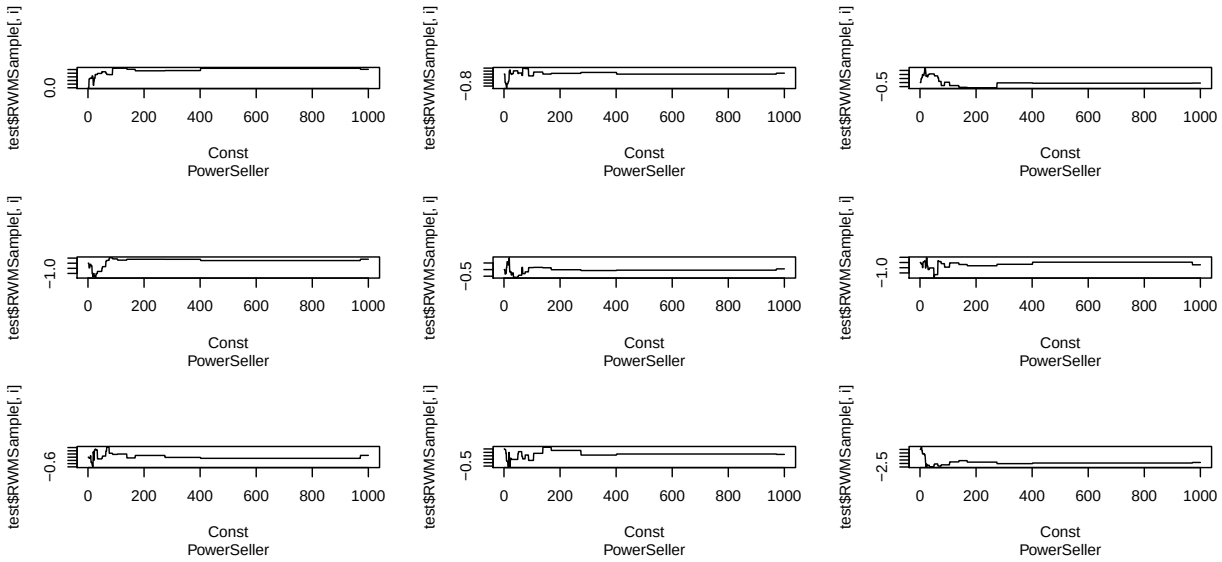
c.

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.*



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)

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(fig.width=9, fig.height = 4.1)
library(tidyverse)
library(dplyr)
library(knitr)
library(mvtnorm)
library(MASS)
set.seed(12345)

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] = 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')
```

```r
#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)
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    ################################################
```

```r
#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)

#----------------------
# 2a.
data0 <- read.table("eBayNumberOfBidderData.dat",header = T)
Y<-data0$nBids
X<-as.matrix(data0[,3:10])

reg_model <- glm(Y ~ X, family = poisson(link = "log"))
summary(reg_model)
# -------------------------
# Q2b.

# setting initial values
y <- as.vector(data0[,1])
X <- as.matrix(data0[,2:length(data0[1,])])
nCov <- dim(X)[2]
covNames <- names(data0)[2:length(data0[1,])]

# Prior
mu <- as.vector(rep(0,nCov))
sigma <- as.matrix(100*solve((t(X)%*%X)))

set.seed(12345)
# Logistic regression function that returns the regression coefficients
logiPost <- function(betas,y,X,sigma){
  pred <- as.vector(X%*%betas)
  lambda0 <- t(X)*betas
  loglike <- sum(y*pred-exp(pred)-log(factorial(y)))
  logprior <- dmvnorm(betas, mean=rep(0,length(betas)), sigma, log=T)
  return(loglike+logprior)
}

# setting initial values
initVal <- as.vector(rep(0,nCov))
# optimize over the betas
optRes <- optim(initVal,logiPost,gr=NULL,y,X,sigma,method="BFGS",control=list(fnscale=-1),hessian=T)

# retrieving betas
beta_hat <- optRes$par
```

```r
beta_hes <- -solve(optRes$hessian)
beta_std <- as.matrix(sqrt(diag(beta_hes)))

# printing results
colnames(beta_hes) <- covNames
rownames(beta_hes) <- covNames
kable(beta_hes)

kable(data.frame(Verification=reg_model$coefficients,Beta_hat=beta_hat,Beta_std=beta_std))

#----------------------------
# 2c.
set.seed(12345)

# the random walk metropolis algorithm in R
# RWMSampler <- function(N, c=0.25, sigma, logPostFunc, theta, ...){
#   sample <- theta
#   for(i in 1:N){
#     prop <- mvrnorm(n=1, theta, c*as.matrix(sigma))
#     proposal <- logPostFunc(prop, ...)
#     target <- logPostFunc(theta, ...)
#     if(runif(1)<exp(proposal-target)){
#       theta <- prop
#       sample <- rbind(sample,theta)
#     }
#   }
#   return(sample)
# }
#
# c <- 0.1
# test<-RWMSampler(1000, c=c, sigma = sigma, logPostFunc =logiPost, theta = initVal, y, X, sigma)
#
# plot(test[,9])
# abline(h=beta_hat[9])

RWMSampler <- function(N, c=0.25, sigma, logPostFunc, theta, ...){
  sample <- matrix(theta,nrow=N,ncol=length(theta))
  alphas <- c()
  for(i in 2:N){
    prop <- as.vector(mvrnorm(n=1, sample[i-1,], c*as.matrix(sigma)))
    proposal <- logPostFunc(prop, ...)
    target <- logPostFunc(sample[i-1,], ...)
    alpha <- min(1,exp(proposal-target))
    U <- runif(1,min=0,max=1)

    if(U<alpha){
      sample[i,] <- prop
    }else{
      sample[i,] <- sample[i-1,]
    }
    alphas[i] <- alpha
  }
  return(list("RWMSample"=sample,"alphas"=alphas))
```

```
}

c <- 0.1
test<-RWMSampler(1000, c=c, sigma = sigma, logPostFunc =logiPost, theta = initVal, y, X, sigma)

# plot the 9 samples covaraite from MCMC
par(mfrow=c(3,3))
for(i in 1:9){
  plot(test$RWMSample[,i],type="l",xlab=covNames)
}
bidders <- c(Const=1,PowerSeller=1,VerifyID=1,Sealed=1,MinBlem=0,MajBlem=0,LargNeg=0,LogBook=1,MinBidSha

set.seed(12345)
pred_bids <- data.frame()

# for(i in 1:nDraws){
#   posterior_betas <- rmvnorm(1000,mean=test$RWMSampler*bidders)
# }
```