TDDE07 - Lab 3

Pontus Svensson (ponsv690) & Kristian Sikiric (krisi211)

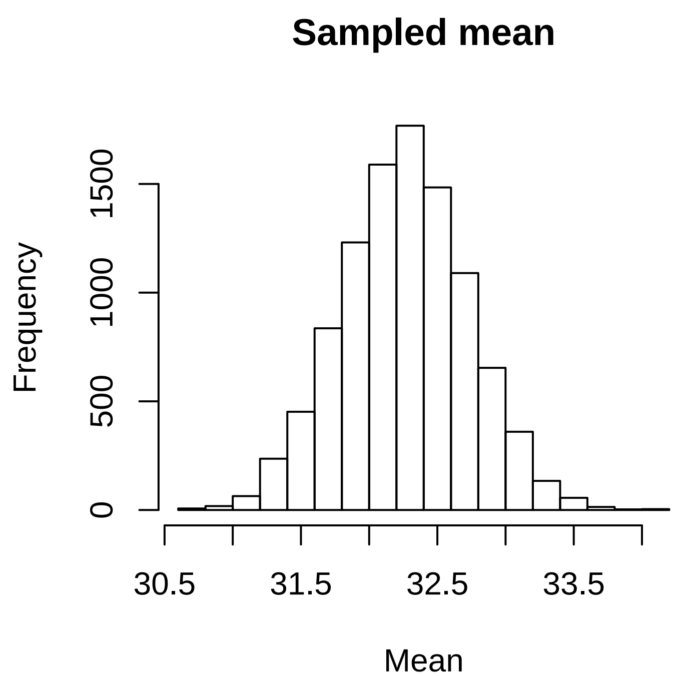
Assignment 1

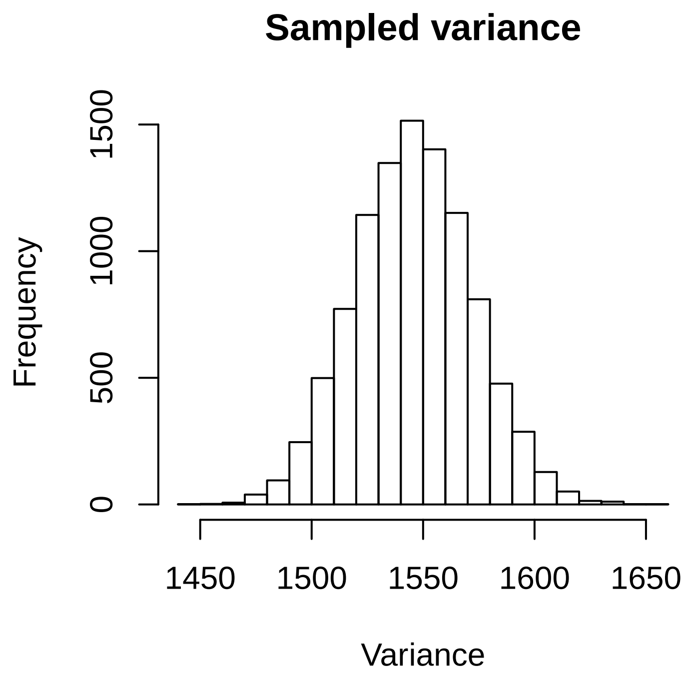
Normal model, mixture of normal model with semi-conjugate prior

In this assignment a data file containing the daily precepitation from year 1948 to 1983 was given. We were to analyze this data using two different models using gibbs sampling. The daily precipitation $y_1, ..., y_n$ were assumed to be independently normal distributed, $y_1, ..y_n|\mu, \sigma^2 \sim \mathcal{N}(\mu, \sigma^2)$ were $\mu \sim \mathcal{N}(\mu_0, \tau_0^2)$ and $\sigma \sim Inv - \mathcal{X}^2(\nu_0, \sigma_0^2)$. Since there are two random variables, we have a bivariate normal distribution.

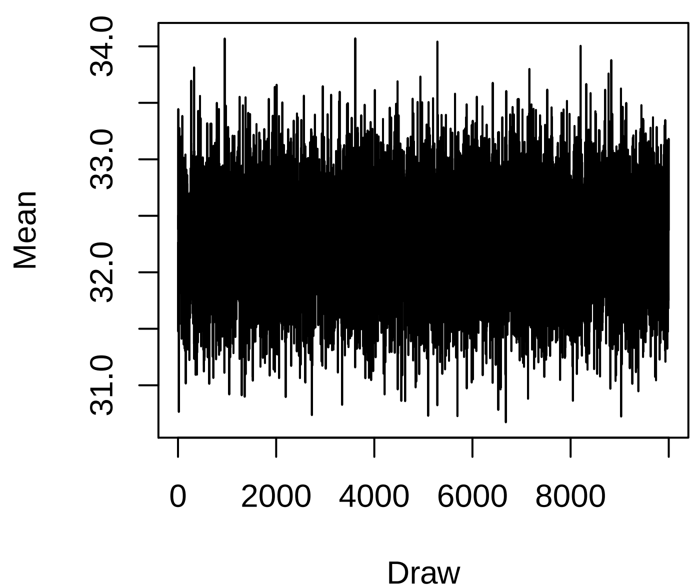
The full conditional posteriors were given to be $\mu|\sigma^2, x \sim \mathcal{N}(\mu_n, \tau_n^2)$ and $\sigma^2|\mu, x \sim Inv{-}\mathcal{X}(\nu_n, \frac{\nu_0\sigma_0^2 + \sum_{i=1}^n (x_i - \mu)^2)}{n + \nu_0})$

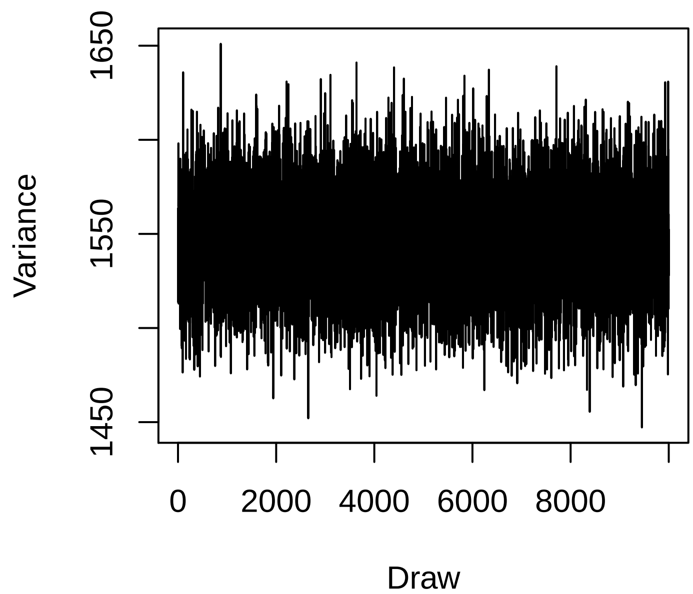
Now we could use these to implement a gibbs sampler, the initial varables were set to some random values based on our priors, (see code). The following histogram shows the result after sampling.

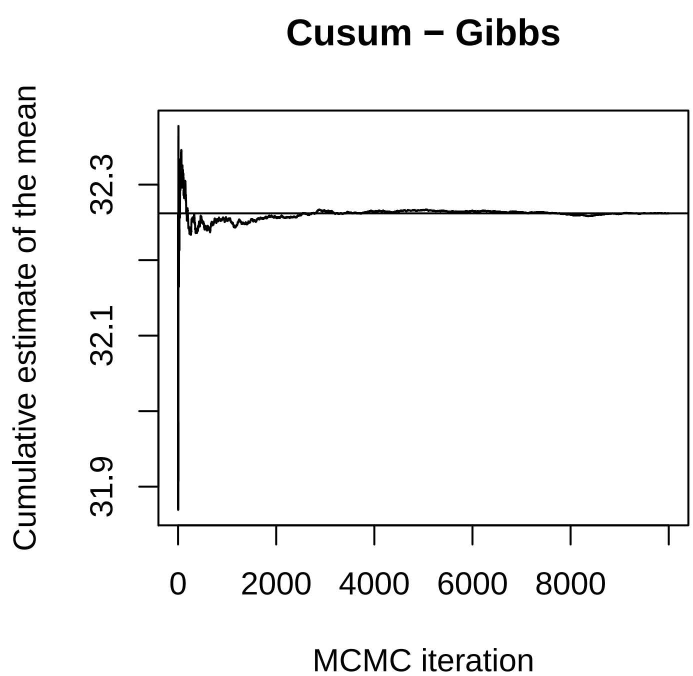


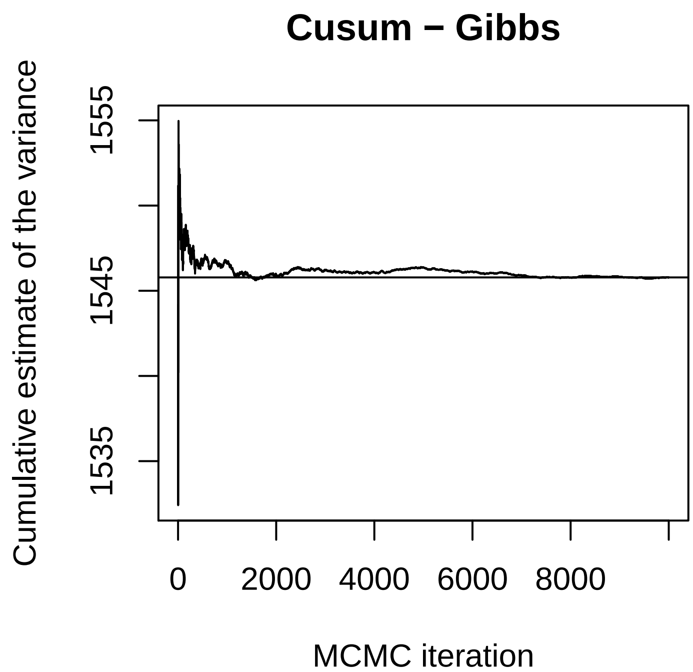


These plots says that the mean precipitation in this period was around $\frac{32.5}{100}$ inches with a variance of $\frac{40}{100}$ inches. The following plots shows the convergence of the sampler.



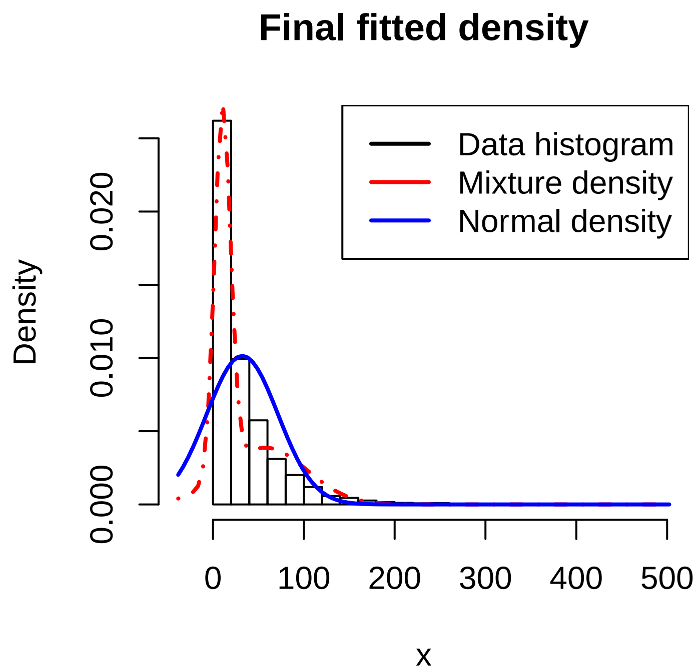

1650
1550
1450
Variance
0
2000
4000
6000
8000
Draw

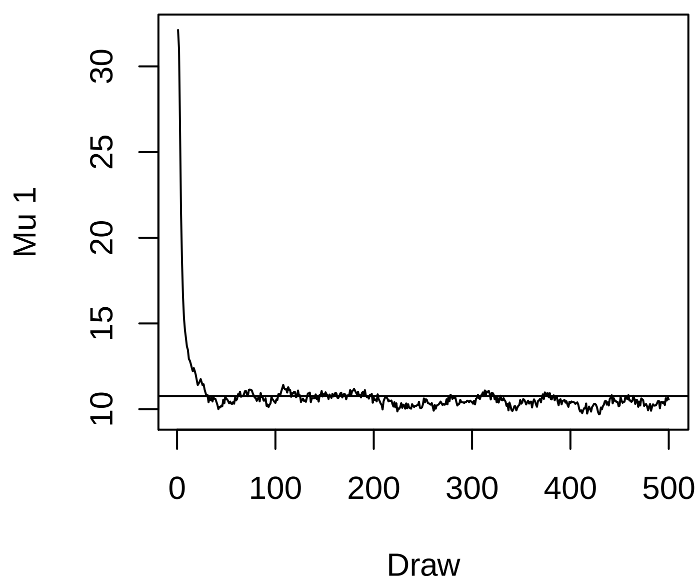

Cusum − Gibbs
32.3
32.1
31.9
Cumulative estimate of the mean
0
2000
4000
6000
8000
MCMC iteration

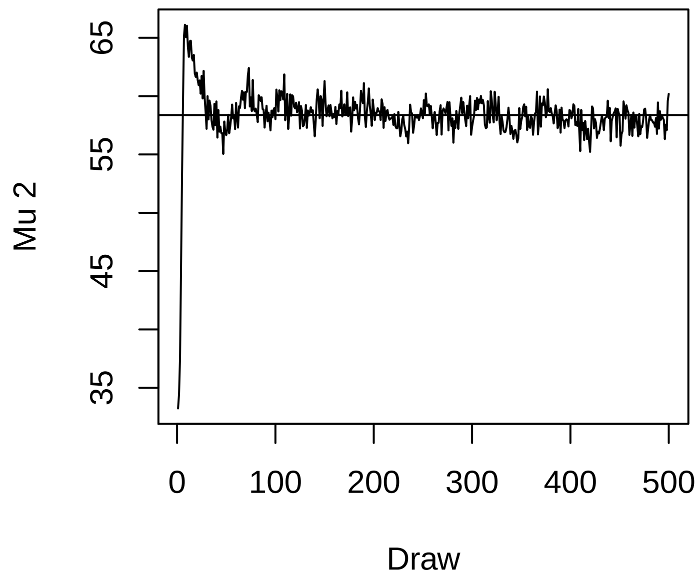


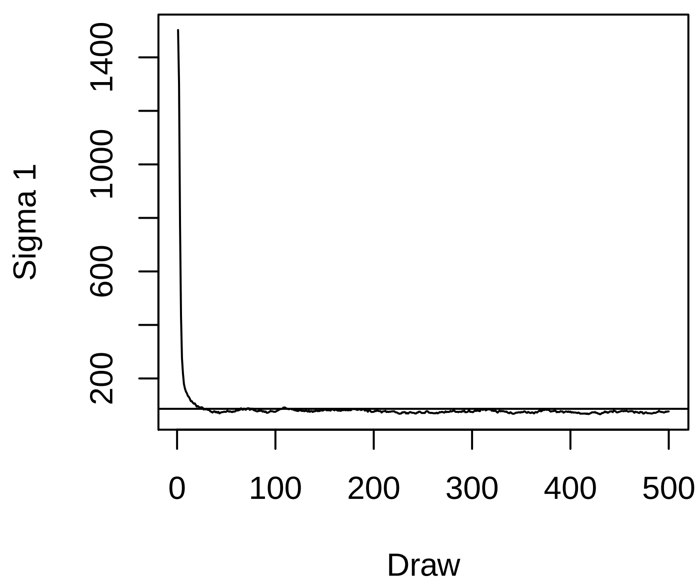
The plots show that the sampler converges. The first two plots of the trajectories shows that the different samples are not so correlated, which is good.

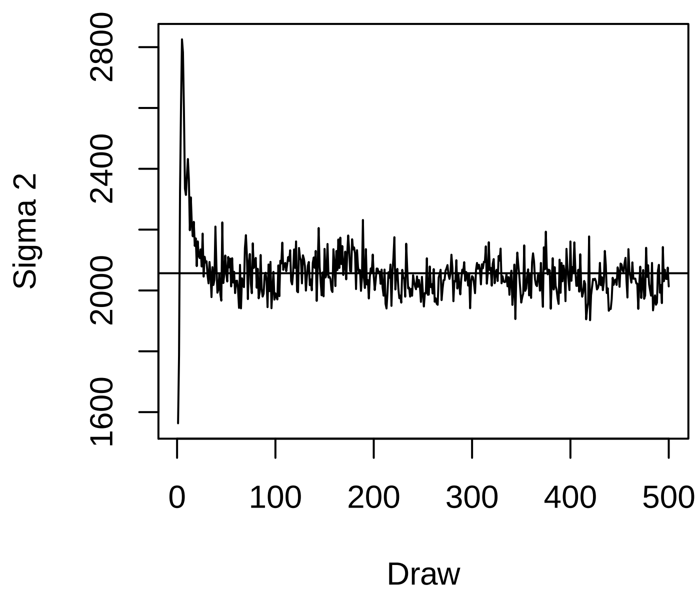
We then repeated the process but now the data followed a two-component mixture of normals model. The code was given, we only extended it so we could show the convergence of the sampler. Below are some plots showing the reults from the sampler with the convergence for the different parameters. We can see that perhaps the first 100 iterations can be seen as the burn-in period.


Final fitted density
Data histogram
Mixture density
Normal density
Density
0.000
0.010
0.020
0
100
200
300
400
500
x

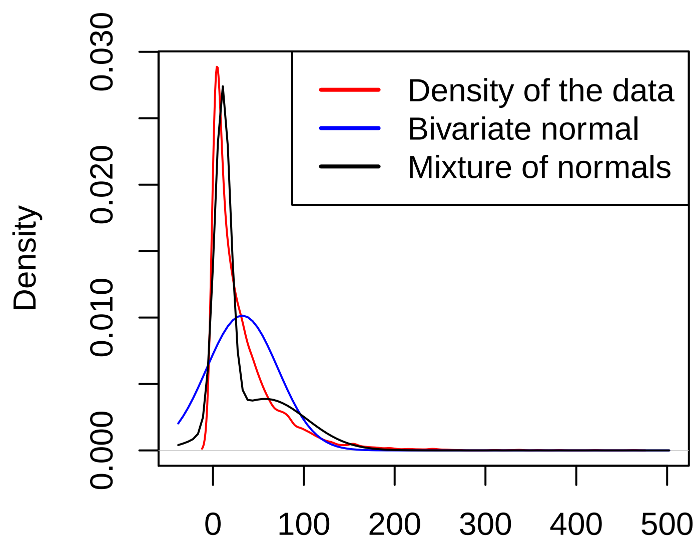

Mu 1
10
15
20
25
30
0
100
200
300
400
500
Draw


65
55
45
35
Mu 2
0
100
200
300
400
500
Draw


1400
1000
600
200
Sigma 1
0
100
200
300
400
500
Draw



Below is a graphical comparison between the denisty of the data, the sampled normal density from above aswell as the sampled mixture of normals.



Assignment 2

Metropolis Random Walk for Poisson regression

In this assignment we considered the following poisson regression model $y_i|\beta \sim Poisson[exp(x_i^T\beta)], i = 1, .., n$ where y_i is the count for the ith observation in the sample and x_p is the p-dimensional vector with covariates. A data set conatining observations from 1000 eBay auctions of coins were given. The target variable in this assignment was the number of bids in each auction.

First a general linear model was fitted using the glm function in R. From this model we could se that the covariates Sealed, VerifyId and MinBidShare were the most significant.

Now we did a Bayesian analysis of the Poisson regression. The prior was $\beta \sim \mathcal{N}[0, 100 * (X^T X)^{-1}]$ where X is the $n \times p$ covariate matrix. The posterior density was assumed to be approximately multivariate normal: $\beta|y \sim \mathcal{N}(\tilde{\beta}, J_y^{-1}(\tilde{\beta}))$. $\tilde{\beta}$ and $J_y(\tilde{\beta})$ were computed with the optim function in R with the log posterior function of the Poisson model. Below the coefficients from the Bayesian analysis with optim is compared to the coefficients of the glm model, as we can see, they are very similar.

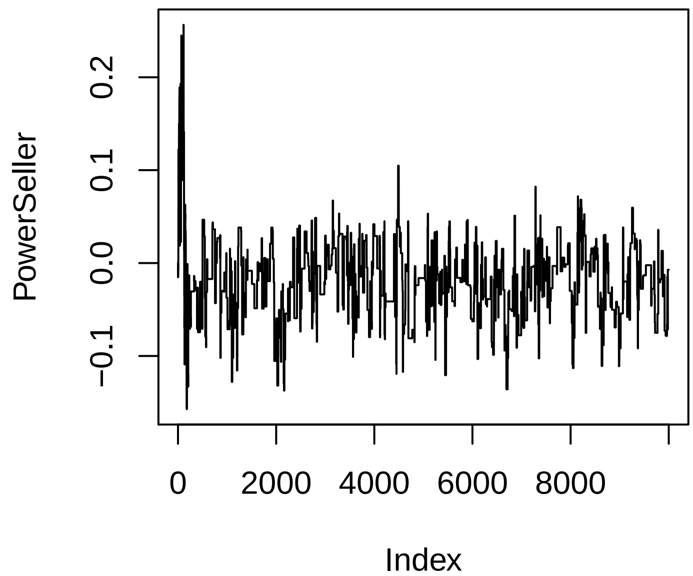
Bayesian analysis

```
## [1]  1.07235230 -0.02033338 -0.39274453  0.44366399 -0.05220897 -0.22145707
## [7]  0.07033797 -0.12041259 -1.89392696
```

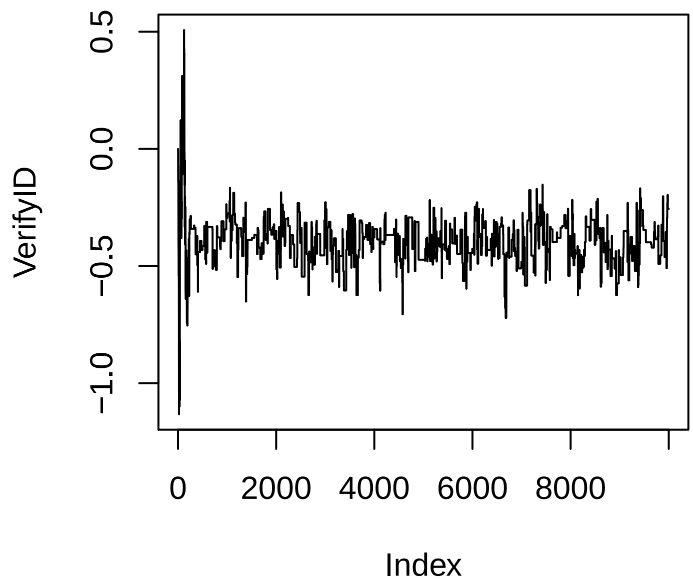
GLM model

```
##       Const PowerSeller    VerifyID      Sealed     Minblem     MajBlem
##  1.07244206 -0.02054076 -0.39451647  0.44384257 -0.05219829 -0.22087119
##     LargNeg     LogBook MinBidShare
##  0.07067246 -0.12067761 -1.89409664
```

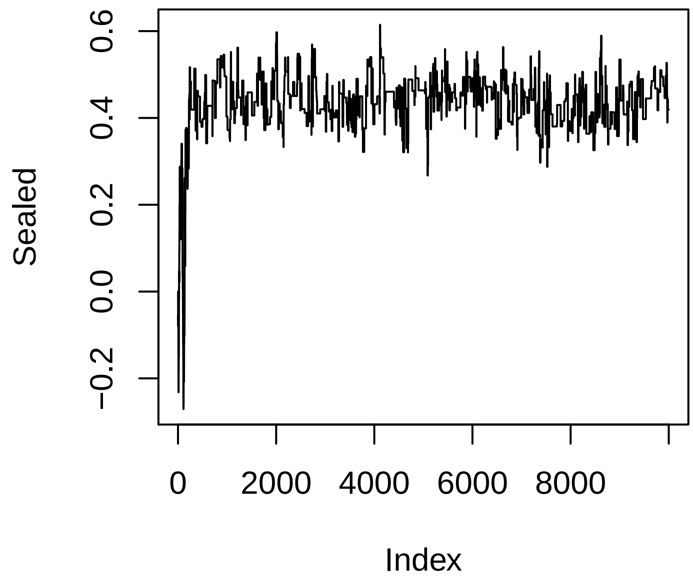
Finally, we simulated from the actual posterior of β using the Metropolis algorithm. Our proposal density was $\theta_p|\theta^{(i-1)} \sim \mathcal{N}(\theta^{(i-1)}, c\sum)$, where $\sum = J_y^{-1}(\tilde{\beta})$ was obtained from the optim function. The value c is a tuning parameter, we used $c = 2$ in this assignment. When simulating with the metropolis algorithm using the same log posterior mentioned above, we got the following convergence curves for the different covariates. The samples does seem a bit corelated, but the result seems good still.

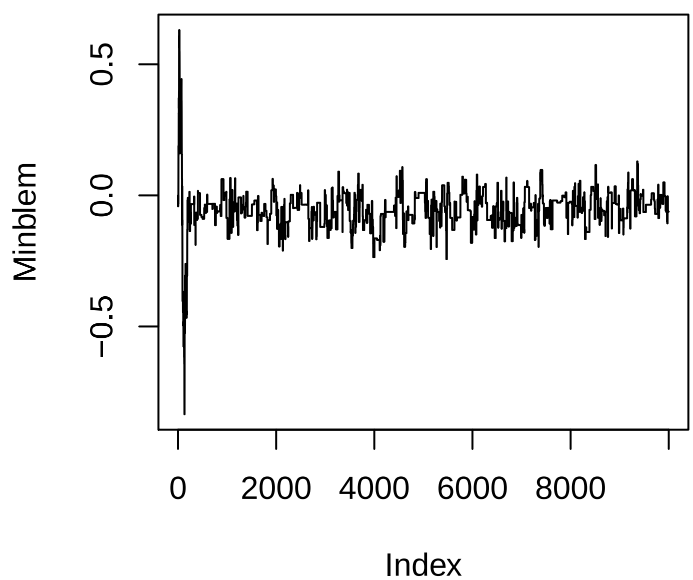


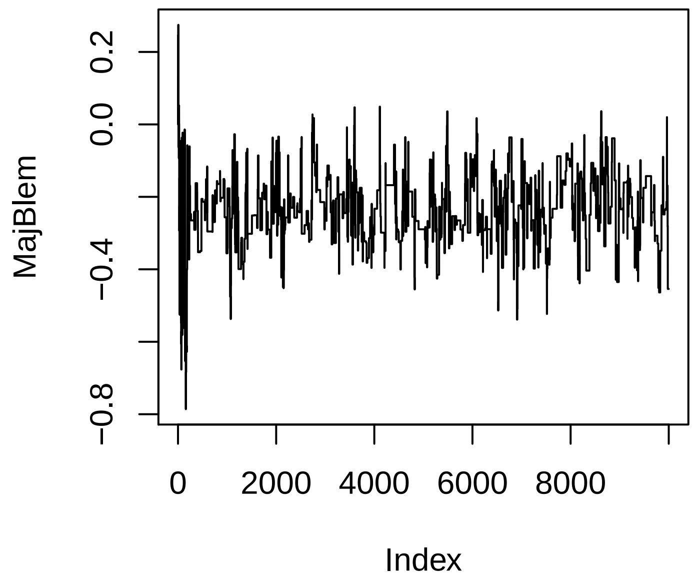
PowerSeller
0.2
0.1
0.0
−0.1
0
2000
4000
6000
8000
Index

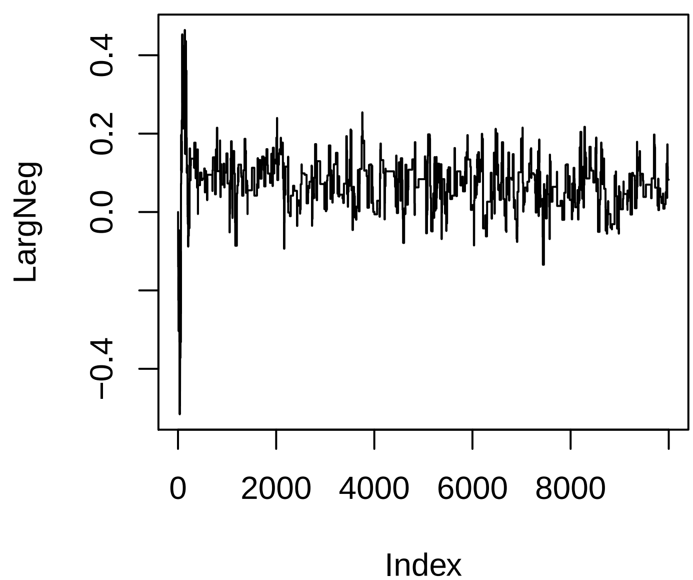


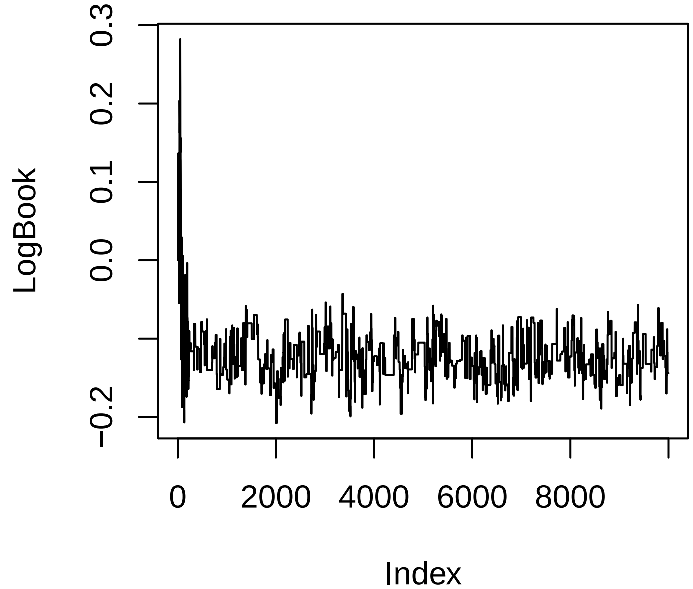
VerifyID
0.5
0.0
−0.5
−1.0
0
2000
4000
6000
8000
Index

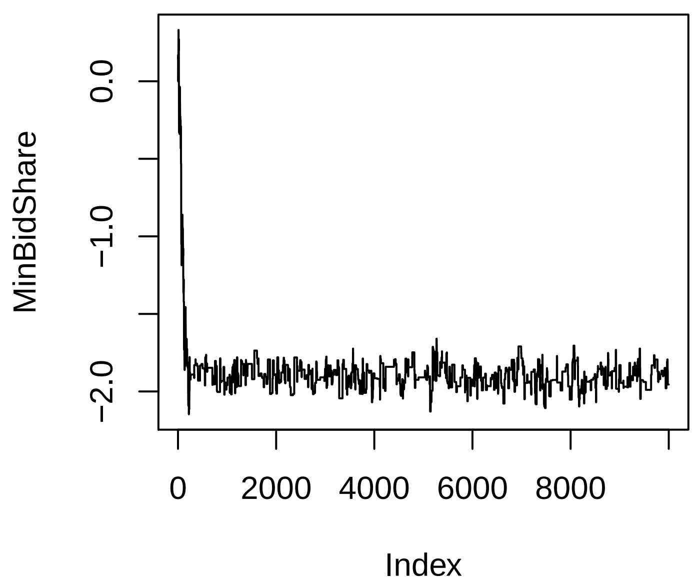

Sealed
0.6
0.4
0.2
0.0
−0.2
0
2000
4000
6000
8000
Index


Minblem
0.5
0.0
−0.5
0
2000
4000
6000
8000
Index


MajBlem
0.2
0.0
−0.4
−0.8
0
2000
4000
6000
8000
Index


LargNeg
0.4
0.2
0.0
−0.4
0
2000
4000
6000
8000
Index





Here we also notice that perhaps the first 1000 iterations can be seen as the burn-in.

And we got the following values for the coefficients, which does not differ much from when using optim in the bayesian analsys.

Metropolis algorithm

```
## [1]  1.06104095 -0.01993666 -0.39559328  0.43507861 -0.05283118 -0.23656841
## [7]  0.07295384 -0.12145117 -1.88318986
```

Appendix

Assignment 1

```
data = read.delim("/Users/kristiansikiric/Desktop/TDDE07/Lab3/rainfall.dat")
#data =read.delim("/home/krisi211/Desktop/TDDE07/Lab3/rainfall.dat")
set.seed(123)
## a)
# Init values
x = mean(data[,1])
n = length(data[,1])

###### Random values #######
mu_0 = x
tau_0 = 10
nu_0 = 3
sigma_0 = 1
sigma = 1 #Init sigma to some value not zero
############################

nu_n = nu_0 + n
NDraws = 10000

#Gibbs sampling
gibbsDraws = matrix(0,NDraws,2)
for( i in 1:NDraws){

  #####FROM LECTURE 2#######
  w = (n/sigma) / (n/sigma + 1/tau_0)
  mu_n = w*x + (1-w)*mu_0
  tau_n = 1/((n/sigma) + (1/tau_0))
  ##########################

  mu_gibbs = rnorm(1,mu_n,sqrt(tau_n))
  gibbsDraws[i,1] = mu_gibbs

  tau = (nu_0*sigma_0 + sum((data[,1]-mu_gibbs)^2))/(n+nu_0)
  sigma = ((nu_n-1)*tau)/rchisq(1,nu_n-1)
  gibbsDraws[i,2] = sigma
}

hist(gibbsDraws[,1])
hist(gibbsDraws[,2])

plot(gibbsDraws[,1],type = 'l')
plot(gibbsDraws[,2],type = 'l')
```

```r
cusumData =  cumsum(gibbsDraws[,1])/seq(1,NDraws)
plot(1:NDraws, cusumData, type = "l", ylab='Cumulative estimate',
     xlab = 'MCMC iteration',
     xlim = c(0,NDraws),
     main = 'Cusum - Gibbs')
abline(h = mean(gibbsDraws[,1]))

cusumData =  cumsum(gibbsDraws[,2])/seq(1,NDraws)
plot(1:NDraws, cusumData, type = "l", ylab='Cumulative estimate',
     xlab = 'MCMC iteration',
     xlim = c(0,NDraws),
     main = 'Cusum - Gibbs')
abline(h = mean(gibbsDraws[,2]))

## b)
#source("/home/krisi211/Desktop/TDDE07/Lab3/NormalMixtureGibbs.R")
source("/Users/kristiansikiric/Desktop/TDDE07/Lab3/NormalMixtureGibbs.R")

## c)
plot(density(data[,1]),col='red',xlim = c(xGridMin,xGridMax))
mu_hat = mean(gibbsDraws[,1])
sigma_hat = mean(gibbsDraws[,2])
lines(xGrid,dnorm(xGrid,mu_hat,sqrt(sigma_hat)), col = 'blue')
lines(xGrid,mixDensMean, type = "l", lwd = 2, lty = 4, col = "black")
```

NormalMixtureGibbs.r

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

##########    BEGIN USER INPUT #################
# Data options
rawData <- read.delim("/Users/kristiansikiric/Desktop/TDDE07/Lab3/rainfall.dat")
x <- as.matrix(rawData['X136'])

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

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(mean(x),nComp) # Prior mean of mu
tau2Prior <- rep(10,nComp) # Prior std of mu
sigma2_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)
nu0 <- rep(2,nComp) # degrees of freedom for prior on sigma2

# MCMC options
nIter <- 500 # 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 ###############
```

```r
###### 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)
  }
  # Diving every column of piDraws by the sum of the elements in that column.
  piDraws = piDraws/sum(piDraws)
  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)))
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,plot=FALSE)$density))
print = TRUE

mixDraws = matrix(0,nIter,4)

for (k in 1:nIter){
  #message(paste('Iteration number:',k))
  alloc <- S2alloc(S)
  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))
    mixDraws[k,j] = mu[j]
  }

  # 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 == j] -
                                                 mu[j])^2))/(nu0[j] + nAlloc[j]))
    mixDraws[k,j+2] = sigma2[j]
  }


  # 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 (plotFit && (k%%1 ==0) && print){
    effIterCount <- effIterCount + 1
    #hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax),
    #main = paste("Iteration number",k), ylim = ylim)
    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("topleft", box.lty = 1, legend = c("Data histogram",components, 'Mixture'),
           #col = c("black",lineColors[1:nComp], 'red'), lwd = 2)
    #Sys.sleep(sleepTime)
  }

}
```

```r
hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax),
     main = "Final fitted density")
lines(xGrid, mixDensMean, type = "l", lwd = 2,
      lty = 4, col = "red")
lines(xGrid, dnorm(xGrid, mean = mean(x),
                   sd = apply(x,2,sd)),
      type = "l", lwd = 2, col = "blue")
legend("topright", box.lty = 1,
       legend = c("Data histogram","Mixture density","Normal density"),
       col=c("black","red","blue"), lwd = 2)

#########################    Our Code    ##############################################
plot(mixDraws[,1],type='l')
abline(h = mean(mixDraws[,1]))
plot(mixDraws[,2],type='l')
abline(h = mean(mixDraws[,2]))
plot(mixDraws[,3],type='l')
abline(h = mean(mixDraws[,3]))
plot(mixDraws[,4],type='l')
abline(h = mean(mixDraws[,4]))
```

Assignment 2

```r
#data = read.delim("//Users/kristiansikiric/Desktop/TDDE07/Lab3/eBayNumberOfBidderData.dat", sep = "")
data = read.delim("/home/ponsv690/Documents/TDDE07/Lab3/eBayNumberOfBidderData.dat",sep = "")
set.seed(123)
X = as.matrix(data[,-1])
y = data[1]
## a)
glm.model= glm(nBids ~0+.,data = data, family = poisson)

#Significant covariates: Sealed, VerifyId, MajBlem(Semi), MinBidShare

## b)

mu = matrix(0,dim(X)[2],1)
sigma = 100 * solve(t(X)%*%X)
initVal = rep(0,dim(X)[2])
library("mvtnorm")

logPoisson = function(betas, y,X,mu,sigma){
  logPos = (sum(y*betas%*%t(X) - exp(betas%*%t(X)) - log(factorial(y))))

  if (abs(logPos) == Inf) logPos = -20001
  logPrior = dmvnorm(betas, mu, sigma)
  return(logPos + logPrior)
}

OptimResults = optim(initVal,logPoisson,gr=NULL,y,X,mu,
                     sigma,method=c("BFGS"),control=list(fnscale=-1),hessian=TRUE)
beta.tilde = OptimResults$par
inv.hessian = -solve(OptimResults$hessian)
```

```r
beta = rmvnorm(10000, beta.tilde, inv.hessian)
(colMeans(beta))
coef(glm.model)

## c)
rwm = function(var, LogPost, theta_prev, ...){
  # Step 1
  theta_p = rmvnorm(1,theta_prev,var)

  # Step 2
  post.theta_prev = LogPost(theta_prev,...)
  post.theta_p = LogPost(theta_p,...)
  alpha = min(1, exp(post.theta_p-post.theta_prev))

  # Step 3
  accepted = runif(1,0,1) < alpha

  if(accepted){
    return(list(theta=theta_p, accepted=accepted))
  }
  else {
    return(list(theta=theta_prev, accepted=accepted))
  }
}

mcmc = function(LogPost, ndraws, ncov) {
  c = 2
  var = c * inv.hessian
  beta = matrix(rep(rep(0, ncov), ndraws), ncol = ncov)
  accepted = rep(0, ndraws-1)

  for(i in 2:ndraws) {
    sample = rwm(var, LogPost, beta[i-1,], y, X, mu, sigma)
    beta[i,] = sample$theta
    accepted[i-1] = sample$accepted
  }

  sum(accepted) / (ndraws-1)
  plot(beta[,2], type = 'l')
  plot(beta[,3], type = 'l')
  plot(beta[,4], type = 'l')
  plot(beta[,5], type = 'l')
  plot(beta[,6], type = 'l')
  plot(beta[,7], type = 'l')
  plot(beta[,8], type = 'l')
  plot(beta[,9], type = 'l')


  return(beta)
}
betas = mcmc(logPoisson, 10000, dim(X)[2])
colMeans(betas)
colMeans(beta) #From b)
```