

Lab3

Andreas C Charitos

14 May 2019

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

a)

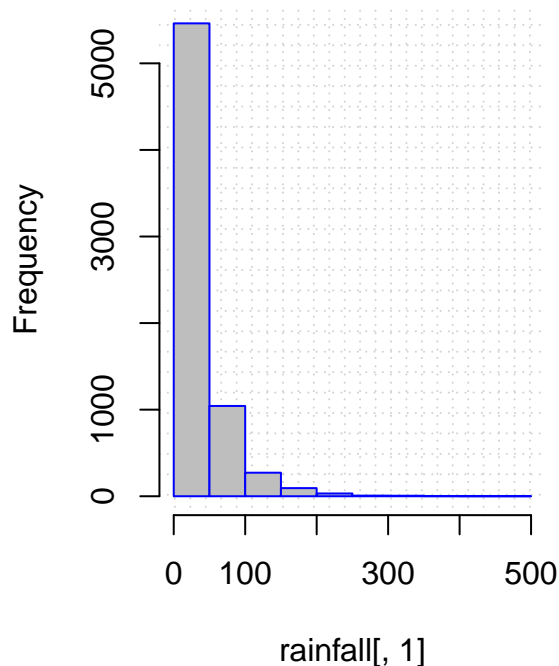
Assume the daily precipitation y_1, \dots, y_n are independent normally distributed, $y_1, \dots, y_n | \mu, \sigma^2 \sim N(\mu, \sigma^2)$ where both μ and σ^2 are unknown. Let $\mu \sim N(\mu_0, \tau_0^2)$ independently of $\sigma^2 \sim \text{Inv} - \chi^2(\nu_0, \sigma_0^2)$

Plot of Histogram and Density

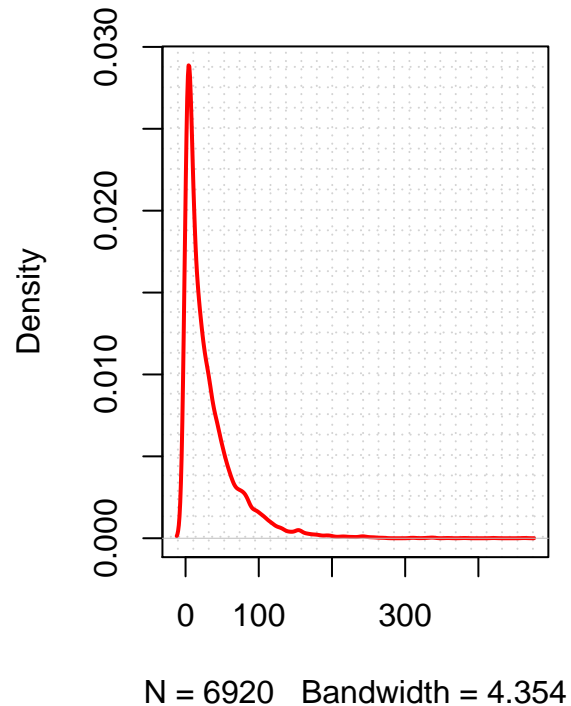
```
# i)
# read data
rainfall<-read.table('rainfall.dat',head=F)

par(mfrow=c(1,2))
hist(rainfall[,1],col="gray",border="blue",
     main="Histogram of Precipitation",panel.first=grid(25,25))
plot(density(rainfall[,1]),col="red",
     main="Density Plot Precipitation",
     lwd=2,panel.first=grid(25,25))
```

Histogram of Precipitation



Density Plot Precipitation



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.

```
# inverse chi-square function
rInvChi2<-function(v0,sigma_sq0){
  # returns one sample from Inv-chi square for given (df,sigma)
  inverse.chi<-(v0*sigma_sq0)/rchisq(1,v0)
  return(inverse.chi)
}

# Gibbs sampler function
GibbsSampler<-function(nIter,mu0,tau0_squared,Sigma0_squared,v0,y){
  n=length(y)
  muPost<-rep(0,nIter) # posterior mu
  SigmaPost<-rep(0,nIter) # posterior sigma
  #ySample<-rep(0,nIter)

  numerator= ( n/Sigma0_squared )+( 1/tau0_squared)
  w=(n/Sigma0_squared)/numerator
  mu_n=w*mean(y)+(1-w)*mu0
  tau_n_squared=1/numerator
  # filling the posterior vectors
  for (i in 1:nIter){

    muPost[i]<-rnorm(n = 1,mean = mu_n,sd = tau_n_squared)
```

```

SigmaPost[i]<-rInvChi2(v0+n, (v0*Sigma0_squared+sum( (y-muPost[i])^2)) / (v0+n) )
#ySample[i]<-rnorm(n=1,muPost[i],SigmaPost[i])
}

return(list('muPost'=muPost,'SigmaPost'=SigmaPost)) #,'samplePost'=ySample))
}
# setting priors
mu0=mean(rainfall[,1])
tau0=1
v0=1
Sigma2=var(rainfall[,1])
# take sample
gibbs_sample=GibbsSampler(1000,mu0,tau0,Sigma2,v0,rainfall[,1] )

```

ii)

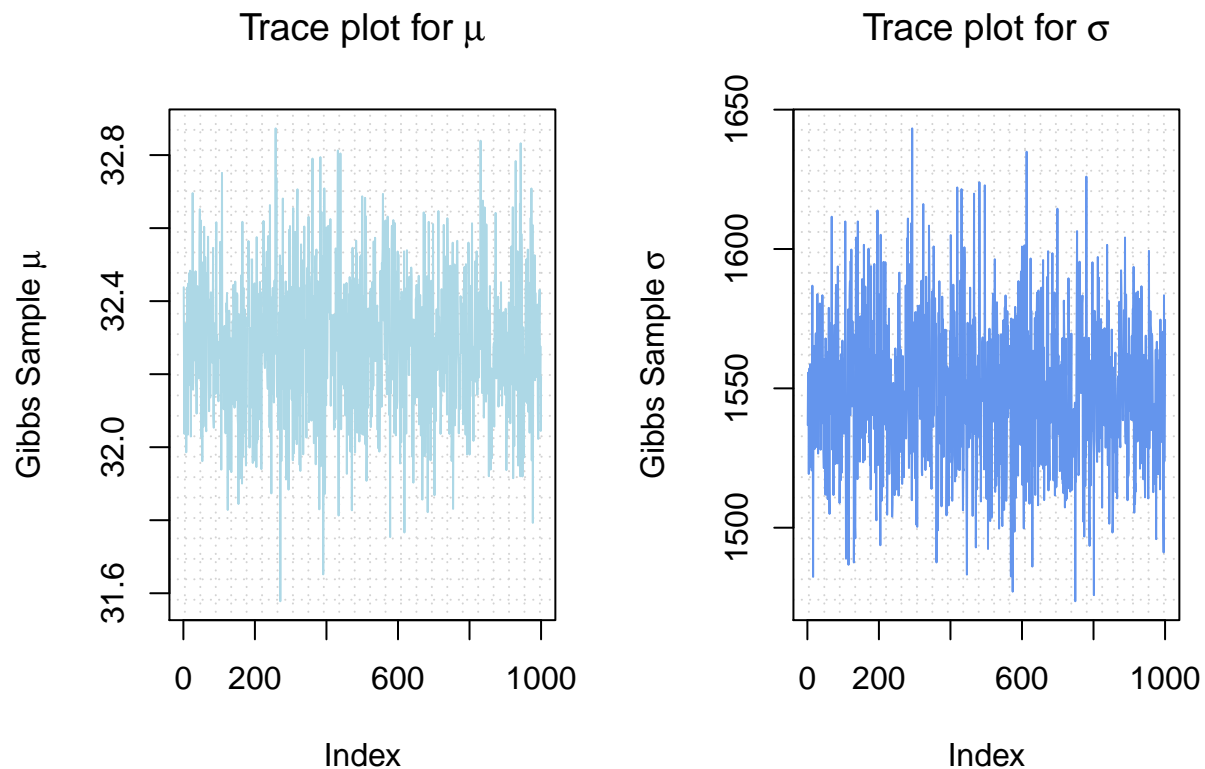
Analyze the daily precipitation using your Gibbs sampler in (a)-i. Evaluate the convergence of the Gibbs sampler by suitable graphical methods, for example by plotting the trajectories of the sampled Markov chains.

Trace Plots

```

par(mfrow=c(1,2))
plot(gibbs_sample$muPost,type = "l",main = expression(paste("Trace plot for ", mu)),
     col="lightblue",ylab=expression(paste("Gibbs Sample ",mu)),
     panel.first = grid(25,25))
plot(gibbs_sample$SigmaPost,type = 'l',
     main = expression(paste("Trace plot for ",sigma)),
     col="cornflowerblue",ylab=expression(paste("Gibbs Sample ",sigma)),
     panel.first = grid(25,25))

```



Converence Plots

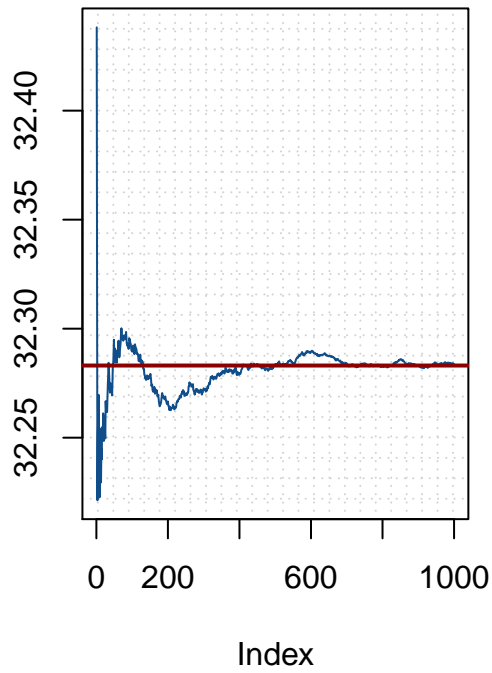
```
library(dplyr)

##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##   filter, lag
##
## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union

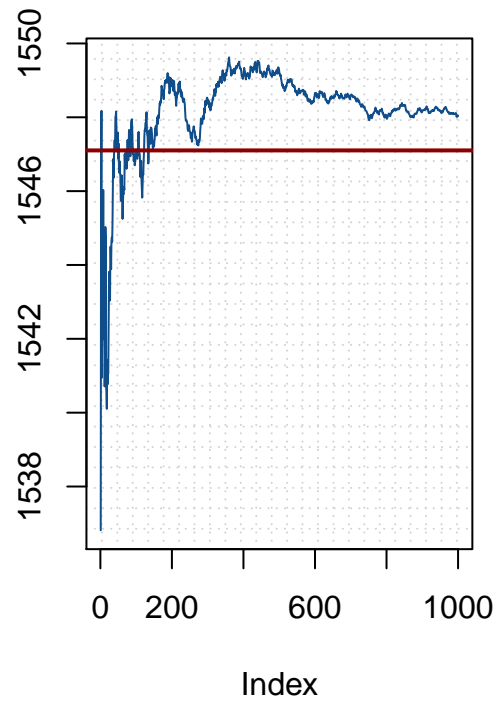
par(mfrow=c(1,2))
plot(cummean(gibbs_sample$muPost),type="l",col="dodgerblue4",
     main=expression(paste("Convergence of ",mu)),ylab = " ",
     panel.first = grid(25,25))
abline(h=mean(rainfall[,1]),col="darkred",lwd=2)

plot(cummean(gibbs_sample$SigmaPost),type="l",col="dodgerblue4",
     main = expression(paste("Convergence of ",sigma)),ylab=" ",
     panel.first = grid(25,25))
abline(h=var(rainfall[,1]),col="darkred",lwd=2)
```

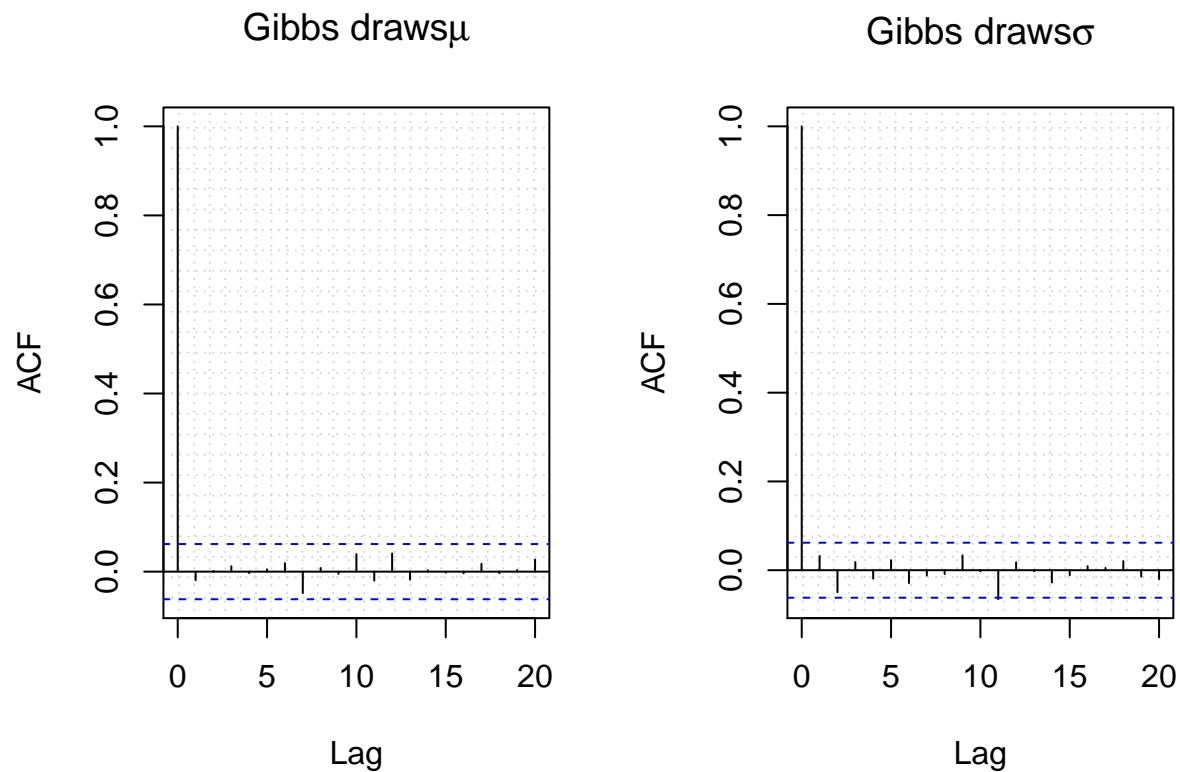
Convergence of μ



Convergence of σ



```
par(mfrow=c(1,2))
acf(gibbs_sample$muPost, main=expression(paste('Gibbs draws',mu)),
    lag.max = 20,panel.first = grid(25,25))
acf(gibbs_sample$SigmaPost, main= expression(paste('Gibbs draws',sigma)),
    lag.max = 20,panel.first = grid(25,25))
```



b)

c)

2. Metropolis Random Walk for Poisson regression

Consider the following Poisson regression model $y_i | \beta \sim \text{Poisson}[\exp(X_i^T \beta)]$, $i = 1, 2, \dots, n$, 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):

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?

```
# a)
# read data
ebay_data=read.table('eBayNumberOfBidderData.dat',head=T)
#dim(ebay_data)
# fit glm model
glmModel <-glm(nBids ~ 0 + ., data = ebay_data, family = poisson(link = "log"))

# create a boolean for the significant coefficients a=0.05
```

```

coeffs_toselect<-summary(glmModel)$coefficients[-1,4]<0.05
# select sig. variables
sign_coeffs<- names(coeffs_toselect)[coeffs_toselect == TRUE]

cat('The significant coefficients are :',sign_coeffs)

```

```
## The significant coefficients are : VerifyID Sealed MajBlem LogBook MinBidShare
```

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 $n \times p$ 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.).

```
# b)
```

```
library("mvtnorm")
```

```
## Warning: package 'mvtnorm' was built under R version 3.5.2
```

```

# Data from the read.table function is a data frame. Let's convert y and X to vector and matrix.
y <- as.vector(ebay_data[,which(names(ebay_data)=='nBids')]);length(y) # response variable

```

```
## [1] 1000
```

```
X <- as.matrix(ebay_data[, -which(names(ebay_data)=='nBids')]);dim(X) # covariates
```

```
## [1] 1000    9
```

```
covNames <- names(X)
```

```
nPara <- dim(X)[2]; # number of covariates
```

```
# setting priors
```

```
tau <- 10;
```

```
Sigma <- tau^2*solve(t(X)%*%X);
```

```
# Logposterior function
```

```
LogPostPoisson <- function(betaVect,y,X,Sigma){
```

```
  nPara <- length(betaVect);
```

```
  linPred <- X%*%betaVect;
```

```
  # evaluating the log-likelihood
```

```
  logLik <- sum( linPred*y -exp(linPred) );
```

```
  if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite, steer the optimizer away from here
```

```
  # evaluating the prior
```

```
  logPrior <- dmvnorm(betaVect, matrix(0,nPara,1), Sigma, log=TRUE);
```

```
  # add the log prior and log-likelihood together to get log posterior
```

```
  return(logLik + logPrior)
```

```

}
# initial value
initVal <- as.vector(rep(0,dim(X)[2]));
# use optim to minimize
OptimResults<-optim(initVal,LogPostPoison,gr=NULL,y,X,Sigma,method=c("BFGS"),control=list(fnscale=-1),h

# Printing the results to the screen
postMode <- OptimResults$par
# Posterior covariance matrix is -inv(Hessian)
postCov <- -solve(OptimResults$hessian)
# Naming the coefficient by covariates
names(postMode) <- covNames
# Computing approximate standard deviations.
approxPostStd <- sqrt(diag(postCov))
# Naming the coefficient by covariates
names(approxPostStd) <- covNames

cat('The posterior mode is:')

## The posterior mode is:
cat("\n")

postMode

## [1] 1.06984118 -0.02051246 -0.39300599 0.44355549 -0.05246627 -0.22123840
## [7] 0.07069683 -0.12021767 -1.89198501

cat("-----")

## -----
cat("\n")

cat('The approximate posterior standard deviation is:')

## The approximate posterior standard deviation is:
cat("\n")

approxPostStd

## [1] 0.03074837 0.03678418 0.09227871 0.05057448 0.06020470 0.09146070
## [7] 0.05634767 0.02895635 0.07109682

```

Plot of posterior covariates

```

library(MASS)
require(RColorBrewer)
# create pallete for sampling
pal<-brewer.pal(9, "Paired")
# take sample
samp<-mvrnorm(1000,postMode,postCov)

par(mfrow=c(2,4))
for(i in 2:9){
  hist(samp[,i],freq=F,

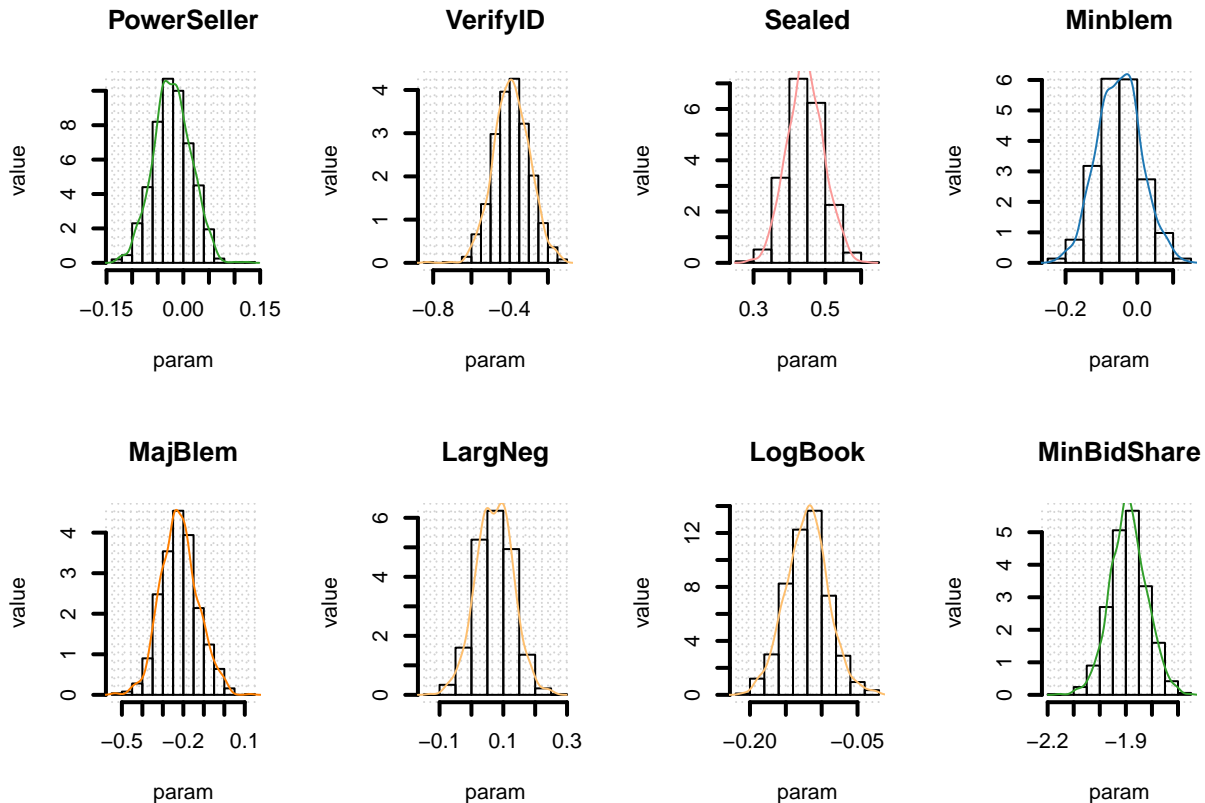
```



```

    main=paste(colnames(X)[i]),
    xlab='param',ylab='value',lwd=2,panel.first = grid(25,25))
    lines(density(samp[,i]),col =sample(pal,1))
}

```



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):

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

where $\Sigma = J_y^{-1}(\tilde{\beta})$ 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.

```

# c)

library(mvtnorm)

```

```

# Metropolis Sampling function
RWMSampler<-function(logPostFunc,n.sim,cx,betaVect,SigmaP,...){
  # initial sample
  initSample=mvrnorm(n = 1,betaVect,SigmaP)
  # matrix to store the samples
  mat_betas<-matrix(nrow=n.sim,ncol=length(initSample))
  # store the initial sample to matrix
  mat_betas[1,<-initSample
  for(i in 2:n.sim){
    # proposal sample
    propSample<-as.vector( mvrnorm(n=1,mat_betas[i-1,],cx*SigmaP) )
    # calculate the approval ratio
    r=exp( logPostFunc(propSample,...)-logPostFunc(as.vector(mat_betas[i-1,]),...) )
    # check the acceptance condition
    if( runif(1)<min(1,r) ){
      mat_betas[i,<-propSample
    }else{
      mat_betas[i,<-mat_betas[i-1,]
    }
  }
  return(mat_betas)
}

# setting the input
betaVect=as.vector(rep(0,dim(X)[2])); y=y; X=X; SigmaP=postCov; n.sim=5000; cx=0.5
res<-RWMSampler(LogPostPoison,n.sim,cx,betaVect,SigmaP,y,X,Sigma) # sample from Metropolis

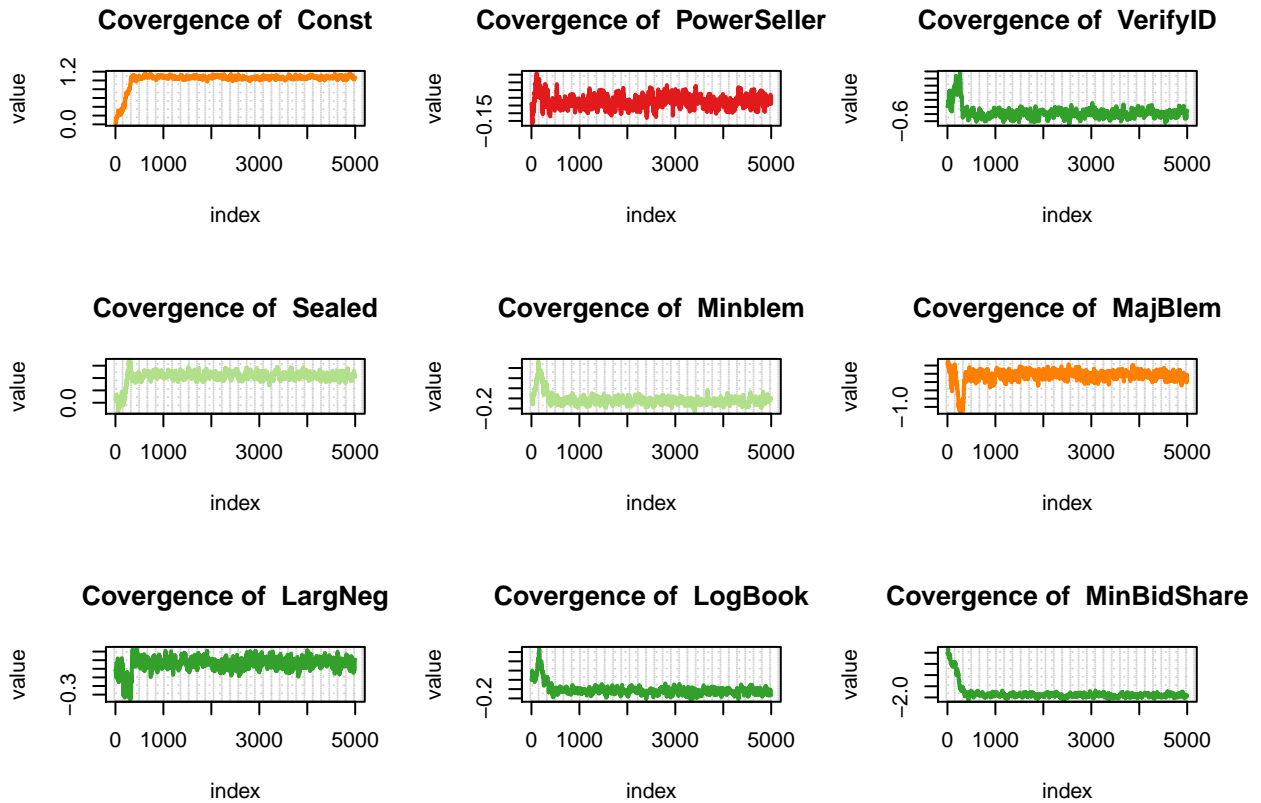
```

Convergence Plots

```

par(mfrow=c(3,3))
for(i in 1:9){
  plot(res[,i],type="l",
    main=paste('Convergence of ',colnames(X)[i]),
    col =sample(pal,1),
    xlab='index',ylab='value',lwd=2,panel.first = grid(25,25))
}

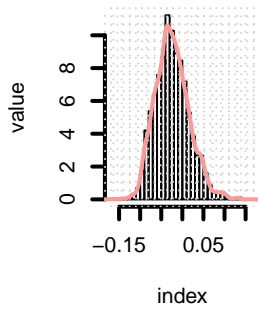
```



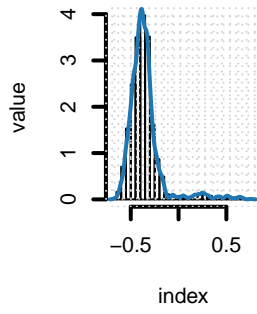
Plot of Histograms and Posteriors

```
par(mfrow=c(2,4))
for(i in 2:9){
  hist(res[,i],freq=F,breaks=30,
       main=paste('Posterior for ',colnames(X)[i]),
       xlab='index',ylab='value',lwd=2)
  lines(density(res[,i]),col =sample(pal,1),lwd=2,panel.first = grid(25,25))
}
```

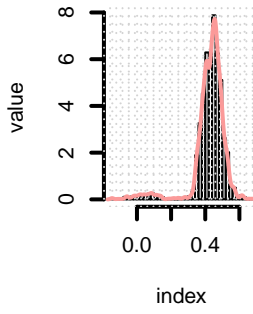
Posterior for PowerSel



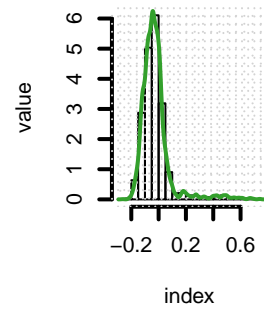
Posterior for VerifyID



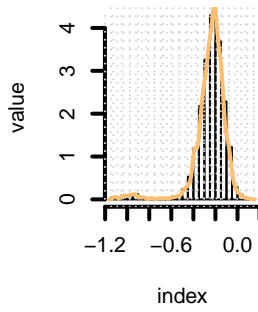
Posterior for Sealed



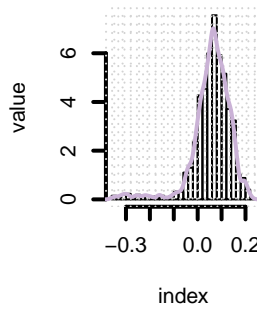
Posterior for Minblen



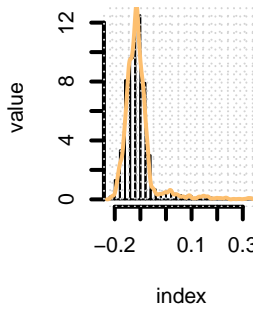
Posterior for MajBlen



Posterior for LargNeq



Posterior for LogBoo



Posterior for MinBidSh.

