GibsSampling-Assignment-6-2

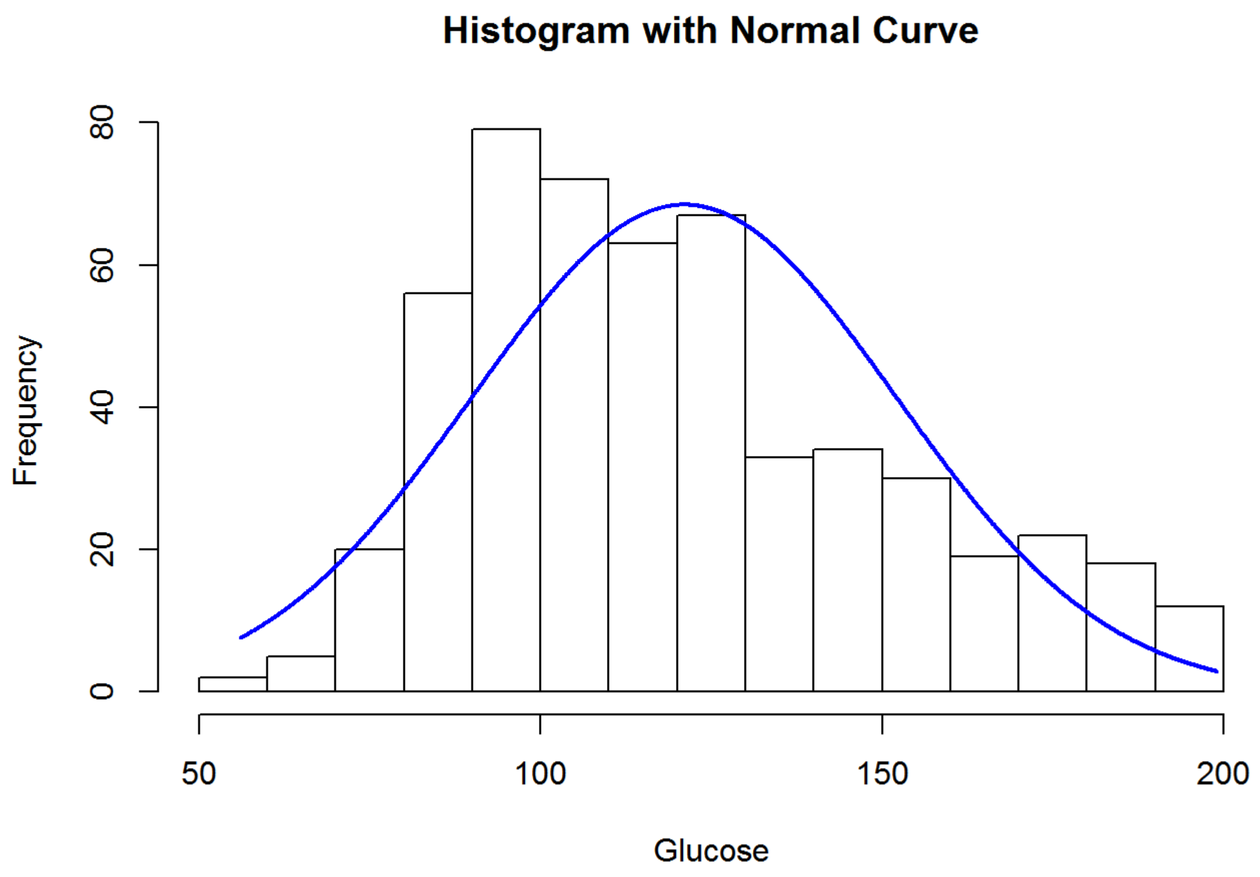
NarenSuri

November 3, 2016

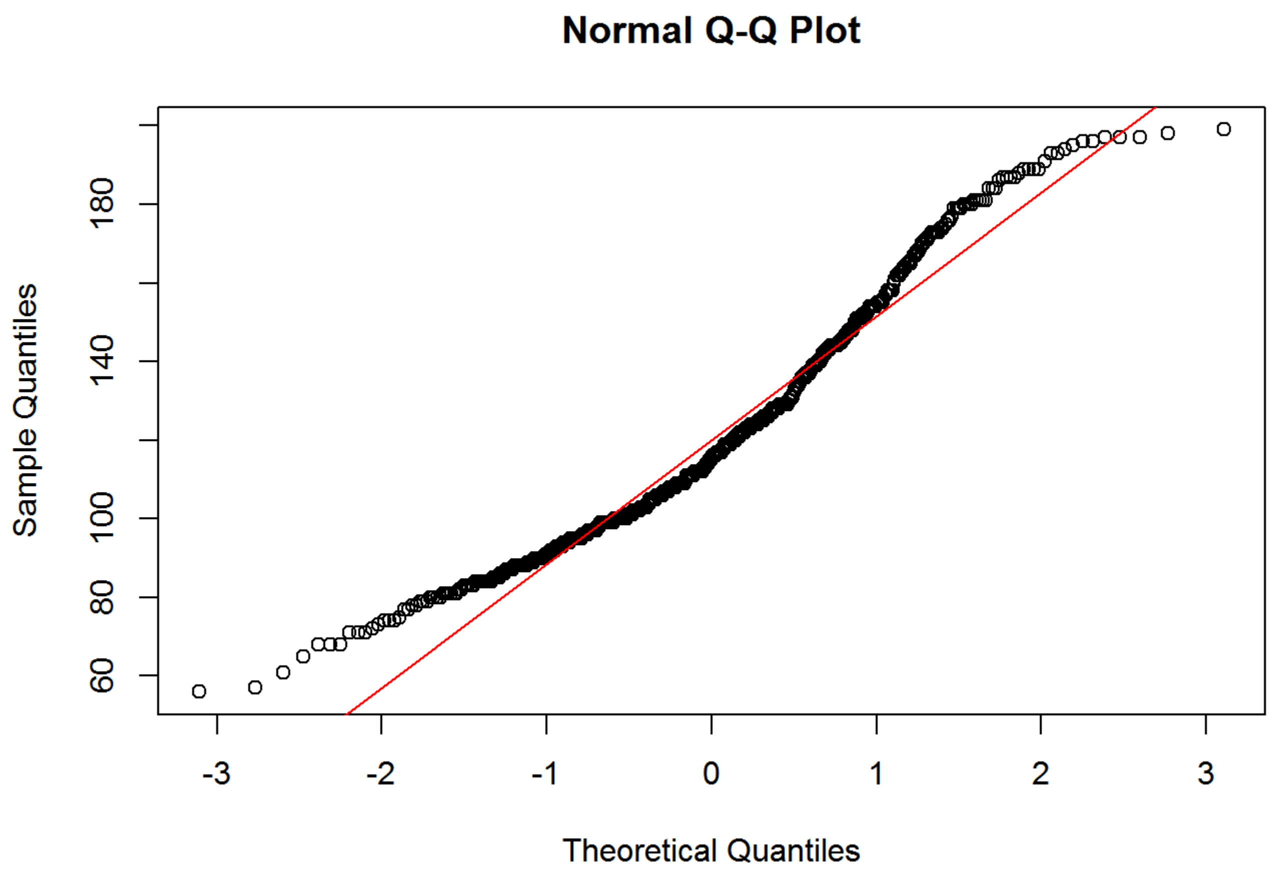
```
ce= read.csv("D:/sem3/Baysien/assignment/assign4/data.csv", header= T,  sep=",")
typeof(ce)
```

```
## [1] "list"
```

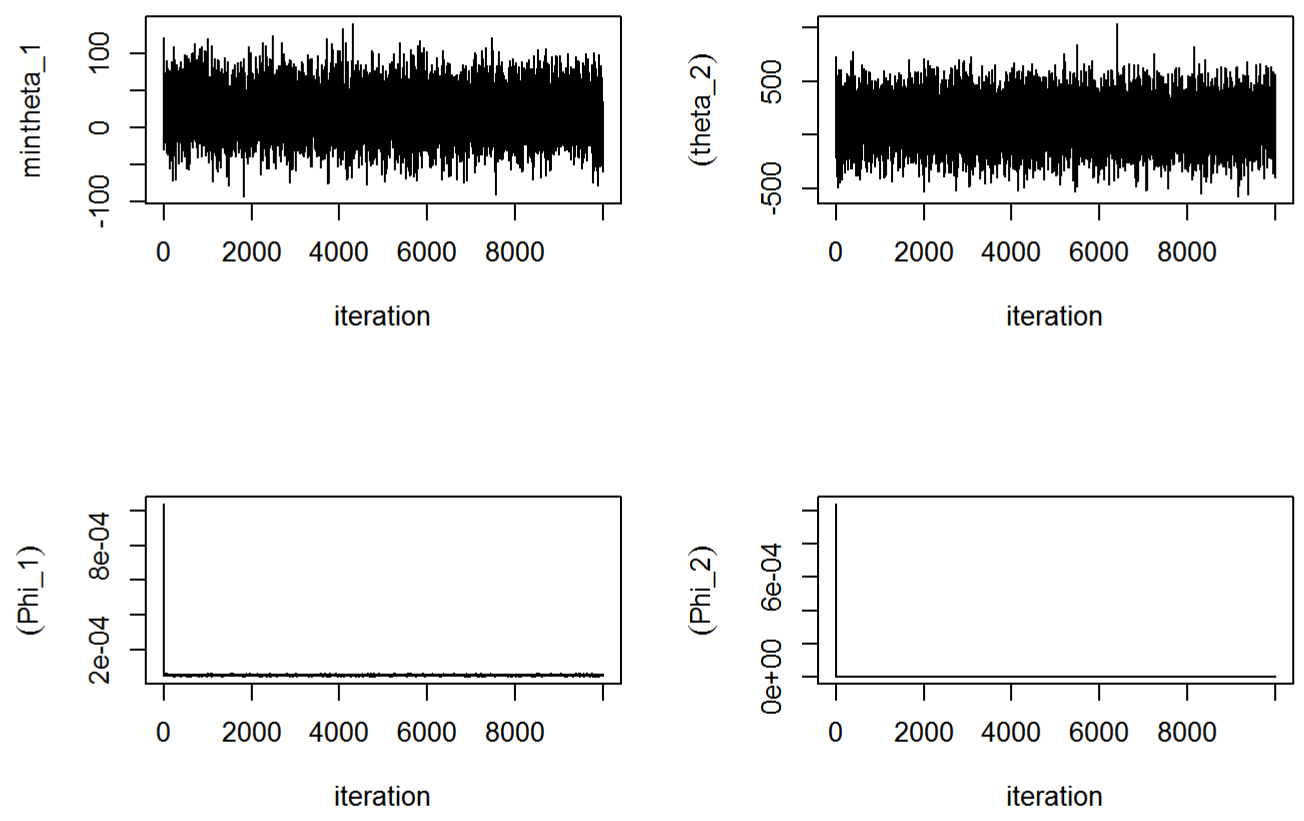
```
xx <- ce$glucose
h<-hist(xx, xlab="Glucose", main="Histogram with Normal Curve")
xfit<-seq(min(xx),max(xx),0.1)
yfit<-dnorm(xfit,mean=mean(xx),sd=sd(xx))
yfit <- yfit*diff(h$mids[1:2])*length(xx)
lines(xfit, yfit, col="blue", lwd=2)
```



```
## Seems like normal, but the data is not completely normal.
qqnorm(ce$glucose)
qqline(ce$glucose, col = 2)
```


Normal Q-Q Plot
Sample Quantiles
60
80
100
140
180
-3
-2
-1
0
1
2
3
Theoretical Quantiles

```
############################################################################
data= read.csv("D:/sem3/Baysien/assignment/assign4/data.csv", header= T,  sep=",")
###Prior values
a=1; b=1
mu0=120;
t0.2= 200
sig0.2 = 1000;
v0=10
S=10000
#################################
p=  matrix(nrow=S, ncol=1)
totalRecords = length(data$glucose)
X= matrix(rep.int(1, totalRecords),nrow=totalRecords, ncol=1)
CurrentP=p[1]=0.3 # initializing the first p value
######
Theta = matrix(nrow=S, ncol=2)
Theta[1,]=c(mean(data$glucose),mean(data$glucose))
Phi = matrix(nrow=S, ncol=2)
Phi[1,]=c(1/var(data$glucose),1/var(data$glucose))
CurrentTheta = c(mean(data$glucose),mean(data$glucose))
CurrentPhi = c(1/var(data$glucose),1/var(data$glucose))
###############################






##################################
PFromBeta <- function(X){
  pval = rbeta(1, a+sum(2-X),b+sum(X-1))
  return(pval)
}

getXi <- function(CurrentP,data,CurrentTheta,CurrentPhi)
{
  H1 = CurrentP * CurrentPhi[1] * exp(-0.5*((data- CurrentTheta[1])^2)* CurrentPhi[1])
  H2 = (1-CurrentP) * CurrentPhi[2] * exp(-0.5*((data- CurrentTheta[2])^2)* CurrentPhi[2])
  pX1 = H1 / (H1+H2)
  pX2 = 1- pX1
  Xi = ifelse(pX1>pX2,1,2)
  return(Xi)

}


######################################

getPhi <- function(data,CurrentTheta,mu0,t0.2,sig0.2,Xi,v0){
  df = data.frame(Xi,data)
  #subsetWithOnes = subset(Xi,Xi==1)
  #length(subsetWithOnes)
  df1 = df[df$Xi==1,]
  n1 = nrow(df1)

  df2 = df[df$Xi==2,]
  n2 = nrow(df2)

  Phi1= rgamma(1,(v0+n1)/2,((v0*sig0.2^2)+sum(df1$glucose-CurrentTheta[1])^2)/2)
  Phi2= rgamma(1,(v0+n2)/2,((v0*sig0.2^2)+sum(df1$glucose-CurrentTheta[2])^2)/2)

  return(c(Phi1,Phi2))


}


#################################################

getThetaVal <- function(data,CurrentPhi,Xi,mu0,t0.2,sig0.2,v0){
  df = data.frame(Xi,data)
  #subsetWithOnes = subset(Xi,Xi==1)
  #length(subsetWithOnes)
  df1 = df[df$Xi==1,]
  n1 = nrow(df1)

  df2 = df[df$Xi==2,]
  n2 = nrow(df2)

  mean1= ifelse(is.nan(mean(df1$data,na.rm=T)),0,mean(df1$data,na.rm=T))
  mean2= ifelse(is.nan(mean(df2$data,na.rm=T)),0,mean(df2$data,na.rm=T))
  a1 = (n1*CurrentPhi[1]*mean1)+(mu0*(1/t0.2))/(n1*CurrentPhi[1]+(1/t0.2))
  b1 = 1/( n1*CurrentPhi[1]+(1/t0.2))
  theta1= rnorm(1,a1,b1)

  a2 = (n2*CurrentPhi[2]*mean2)+(mu0*(1/t0.2))/(n2*CurrentPhi[2]+(1/t0.2))
  b2 = 1/( n2*CurrentPhi[2]+(1/t0.2))
  theta2= rnorm(1,a2,b2)

  return(c(theta1,theta2))
}


################################################################################
# Gibbs sampling technique - conneccting the conditionals
for(i in 2:10000){

  CurrentP=PFromBeta(X)
  p[i,1]= CurrentP

  Xi = getXi(CurrentP,data$glucose,CurrentTheta,CurrentPhi)
  X= Xi

  PhiVals = getPhi(data$glucose,CurrentTheta,mu0,t0.2,sig0.2,X,v0)
  Phi[i,1]=PhiVals[1]; Phi[i,2]=PhiVals[2]
  CurrentPhi = PhiVals

  thetaVals = getThetaVal(data$glucose,CurrentPhi,Xi,mu0,t0.2,sig0.2,v0)
  CurrentTheta= thetaVals
  Theta[i,1]=thetaVals[1];Theta[i,2]=thetaVals[2]

}

################################################
## Plot the theta vs iteration value
#expression(theta)
par(mfrow = c(2,2))
plot(seq(1:10000),Theta[,1] ,xlab="iteration",ylab=expression(min(theta_1)), type='l')
plot(seq(1:10000),Theta[,2] ,xlab="iteration",ylab=expression((theta_2)), type='l')
##
plot(seq(1:10000),Phi[,1] ,xlab="iteration",ylab=expression((Phi_1)), type='l')
plot(seq(1:10000),Phi[,2] ,xlab="iteration",ylab=expression((Phi_2)), type='l')
```



```r
######################################################
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.2.5
```

```r
THetaTheta = data.frame(Theta)
MinOfThetas=mdply(THetaTheta, function(X1,X2){min(X1,X2)})
head(MinOfThetas)
```

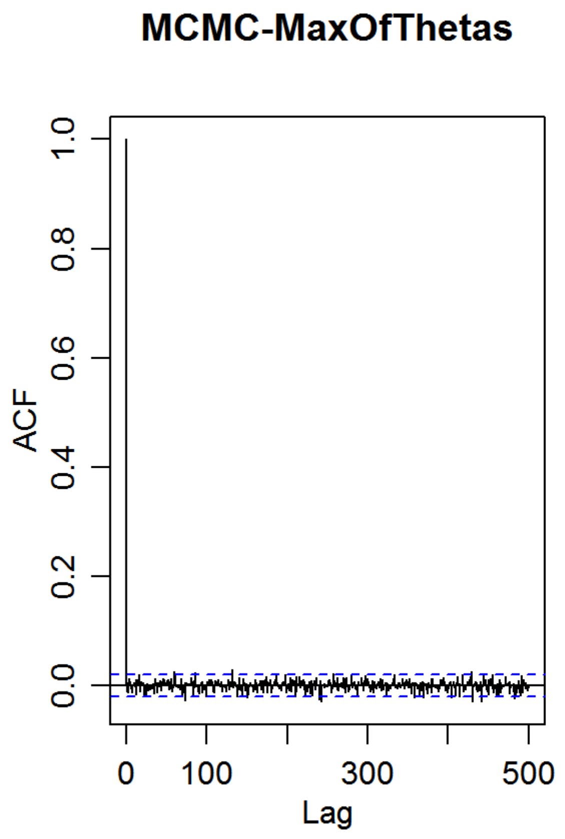
```
##           X1         X2         V1
## 1 121.030075 121.030075 121.030075
## 2  -8.188797 111.948946  -8.188797
## 3  21.636717   3.454418   3.454418
## 4  23.809692 -13.090178 -13.090178
## 5 -29.462303 156.316709 -29.462303
## 6  51.406736 185.791402  51.406736
```

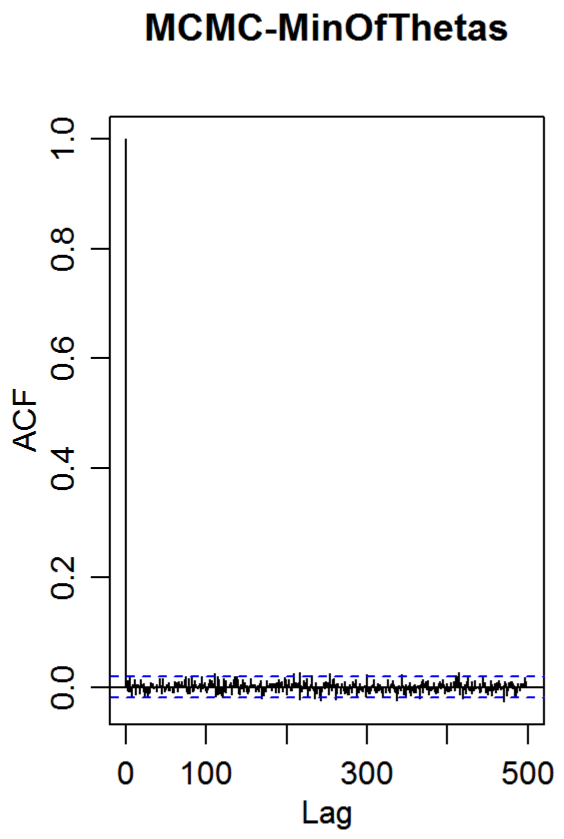
```
MaxOfThetas=mdply(THetaTheta, function(X1,X2){max(X1,X2)})
head(MaxOfThetas)
```

```
##           X1         X2        V1
## 1 121.030075 121.030075 121.03008
## 2  -8.188797 111.948946 111.94895
## 3  21.636717   3.454418  21.63672
## 4  23.809692 -13.090178  23.80969
## 5 -29.462303 156.316709 156.31671
## 6  51.406736 185.791402 185.79140
```

```
#############################################################################
##### All the values shown below are drawn to the min and max Theta values
#############################################################################
par(mfrow=c(1,2),mgp=c(1.75,.75,0))
acf(MaxOfThetas$V1,lag.max=500, main = 'MCMC-MaxOfThetas')
#acf(MaxOfThetas$V1,lag.max=100, main = 'MCMC')
#acf(MaxOfThetas$V1,lag.max=1000, main = 'MCMC')
acf(MinOfThetas$V1,lag.max=500, main = 'MCMC-MinOfThetas')

#Effective sample size
#install.packages("coda")
library(coda)
```

```
## Warning: package 'coda' was built under R version 3.2.5
```





```
effectiveSize(MinOfThetas$V1)
```

```
##  var1
## 10000
```

```
effectiveSize(MaxOfThetas$V1)
```

```
##  var1
## 10000
```

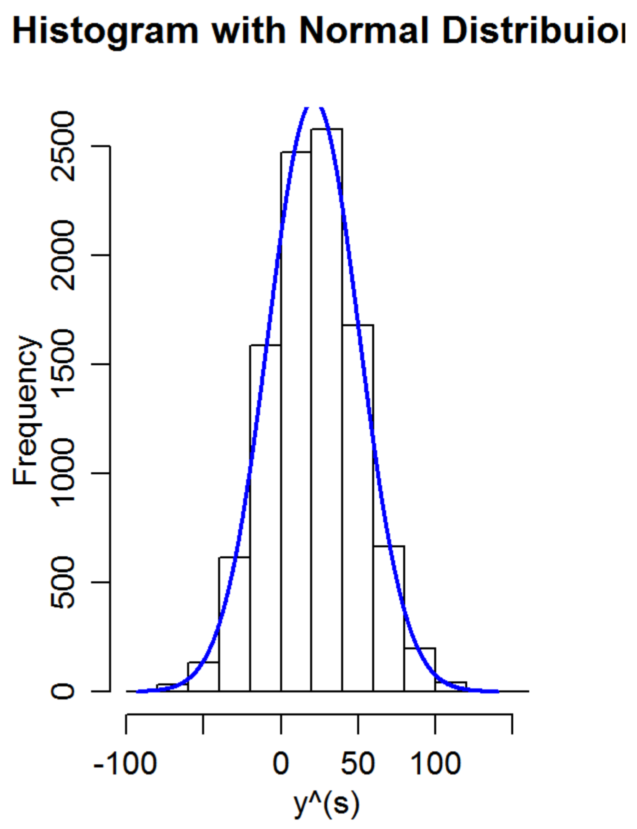
```
#Coda diagnostics
min.mcmc <- mcmc(MinOfThetas$V1)
summary(min.mcmc)
```

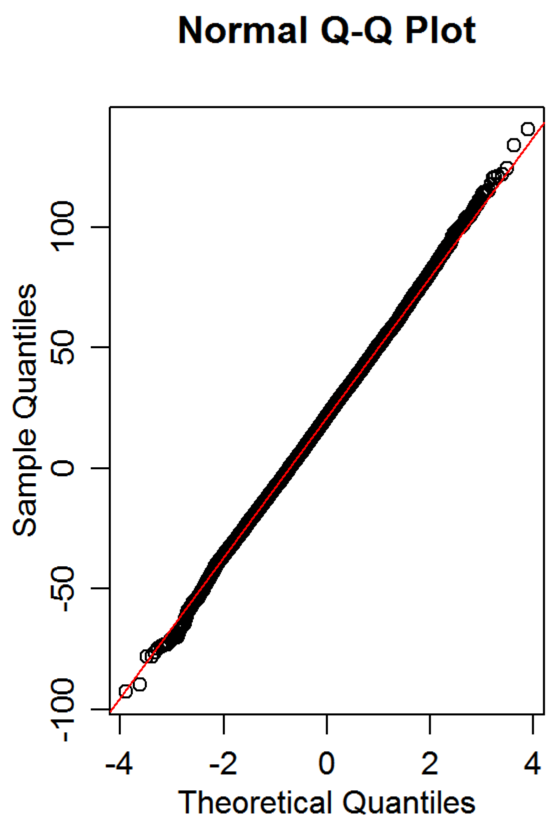
```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##       -19.3439        85.6625         0.8566         0.8566
##
## 2. Quantiles for each variable:
##
##     2.5%      25%      50%      75%    97.5%
## -271.783  -31.260    5.673   30.838   72.331
```

```
max.mcmc <- mcmc(MaxOfThetas$V1)
summary(max.mcmc)
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##        160.043        147.928          1.479          1.479
##
## 2. Quantiles for each variable:
##
##   2.5%    25%    50%    75%  97.5%
## -14.96  39.10 118.12 252.08 514.92
```

```
####################################################

#ths<-seq(-70,130,length=1000) #grid of \theta
# change -5 to -9 for more separated mode
#plot(ths,  ,type="l", ylab = expression(p(theta)), xlab = expression(theta))

##################################################################################

###### 6.2 D
## We have Thetas and Sigmas and Xi Vector
## lets sample each one from the p vector
PredX = matrix(rep.int(1, S),nrow=S, ncol=1)
PredY = matrix(rep.int(1, S),nrow=S, ncol=1)
for (k in 1:S){
   FromWhichDistrib= ifelse(p[k,1]>=0.5,1,2)
   PredX[k,1]=FromWhichDistrib
   PredY[k,1]=rnorm(1,Theta[k,FromWhichDistrib],Phi[k,FromWhichDistrib])
}

h <- hist(PredY[,1], xlab="y^(s)", main="Histogram with Normal Distribuion")
xfit<-seq(min(PredY[,1]),max(PredY[,1]),0.1)
yfit<-dnorm(xfit,mean=mean(PredY[,1]),sd=sd(PredY[,1]))
yfit <- yfit*diff(h$mids[1:2])*length(PredY[,1])
lines(xfit, yfit, col="blue", lwd=2)
## Seems like normal, but the data is not completely normal.
qqnorm(PredY[,1])
qqline(PredY[,1], col = 2)
```





```
# this is having a perfect normal shape.
# the resultant PredX has all 1's mostly and this says that the Xi==1 most of the times
# This says that most of the data is coming from the single ditribution, and it looks like this data is not actually a mixtu
re model based.
# and the resultant his is perfectly distributed.
```