732A91: Lab 3
Bayesian Learning

Sarah Alsaadi, Carles Sans Fuentes

May 22, 2017

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

The data rainfall.dat consists of daily records, from the beginning of 1948 to the end of 1983, of precipitation (rain or snow in units of 1 inch, and records of zero 100 precipitation are excluded) at Snoqualmie Falls, Washington. Analyze the data using the following two models.

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

   (a) Implement (code!) a Gibbs sampler that simulates from the joint posterior $p(\mu, \sigma^2|y_1, \ldots, y_n)$. Where the full conditional posteriors are given by:

      i. $\mu|\sigma^2, y_1, \ldots, y_n \sim N(\mu_n, \tau_n^2)$ where $\mu_n = \frac{n/\sigma^2}{n/\sigma^2+1/\tau_0^2}\bar{y} + \frac{1/\tau_0^2}{n/\sigma^2+1/\tau_0^2}\mu_0$ and $\tau_n^2 = \frac{1}{n/\sigma^2+1/\tau_0^2}$

      ii. $\sigma^2|\mu, y_1, \ldots, y_n \sim Inv - \chi^2(v_0 + n, \frac{v_0\sigma_0^2+\sum_{i=1}^n(y_i-\mu)^2)}{n+v_0}$.

   The initial values have been set up near to 0 but τ, which has been chosen as 10. Proving with different τ values showed us that small τ, which accounts for variance, can lead our distribution of posterior estimates to stuck in local minimas since it is not able to get out from an optima. Still, only results with $\tau = 10$ will be shown, but it is interesting to know that with an smaller τ our optimal posterior μ distribution was lower (at around 26-28).

   (b) Here below in figure 1 it can be seen how the posterior of our μ and σ converges to 31-33 and 1550 respectively. There is a burning period for$\sigma and \mu$ to converge to the distribution.

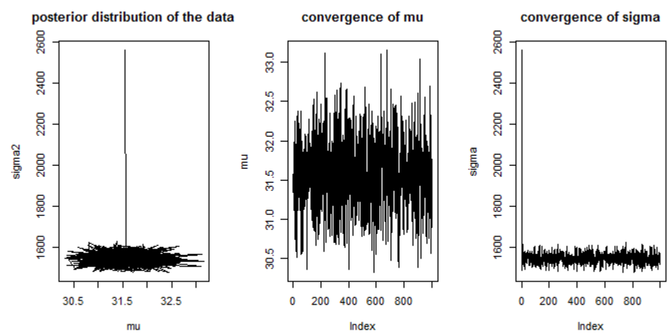

Figure 1: 3 plots of the posterior distribution of $\mu and \sigma$ with 1000 draws

In 2, we can see how the μ parameter is distributed with its cumulative mean that goes to 32 and how data is correlated. It can be seen that data is correlated just if we take each or every 2 observations. Thus, every third answer should be taken because it does not show any more autocorrelation or dependence.

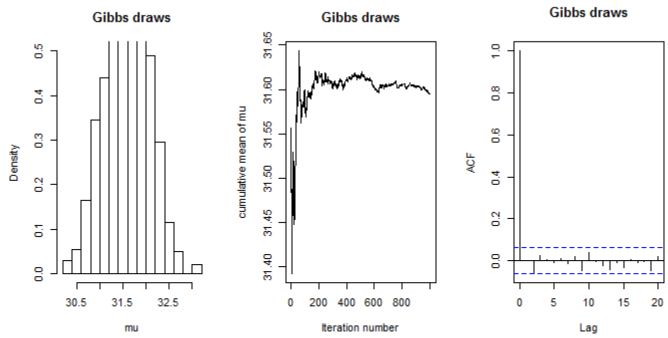

Figure 2: distribution of μ with 1000 draws,cumulative mean of the mean of μ and correlation lags

(c) In 1b, we are asked to run a mixture of normal on the data for 2 μ and σ, assuming that data could come from two different points in Sweden.

In the following figure 3 below it can be seen the convergence of the mixture density model after 100 iterations.

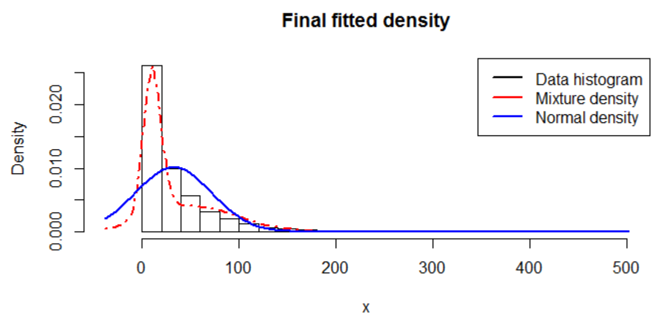

Figure 3: Convergence of the mixture density model after 100 iterations

Now (in 1c) we are asked to plot in one figure 4 to 1) a histogram or kernel density estimate of the data, 2) Normal density $N(\mu, \sigma^2)$ in (a) and 3) Mixture of normals in(b). This is done in the following graph being the red line the density 1000 random sample numbers from the distribution

$$N(\mu = mean(\mu_1), sd = sqrt(mean(sigma2)))$$

and the yellow one a mixture from activity b also from a 1000 random generated data from 1000 points with from

$$\pi * N(\mu_1, \sigma_1^2) + (1 - \pi) * N(\mu_2, \sigma_2^2)$$

.

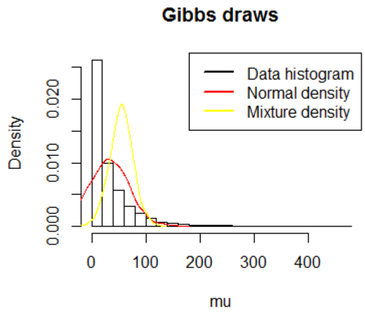

Figure 4: a histogram or kernel density estimate of the data, 2) Normal density $N(\mu, \sigma^2)$ in (a) and 3) Mixture of normals in(b)

Results show that none of both distributions is good for modeling the data. It is not a good idea just to take one or two normal distributions to model exponential data. More distributions would be necessary to get a better model.

Probit regression

1. Implement (code!) a data augmentation Gibbs sampler for the probit regression model

$$Pr(y = 1|\mathbf{x}) = \mathbf{\Phi}(\mathbf{x^T}\beta)$$

   The code is given in the appendix.

2. Compute the posterior of β in the probit regression for the WomenWork dataset from Lab 2 using the prior $\beta \sim N(0, \tau^2 I)$, with $\tau = 10$.

3. Do a normal approximation $\beta|\mathbf{y}, \mathbf{X} \sim \mathbf{N}(\tilde{\beta}, \mathbf{J_y^{-1}}(\tilde{\beta}))$ of the posterior for β in the probit regression. Compare with the results from 2(b). Is the normal approximation accurate? In figure 5 and 6, you can see the distribution of our parameters. Also the the mean is provided with its 95% intervals below:

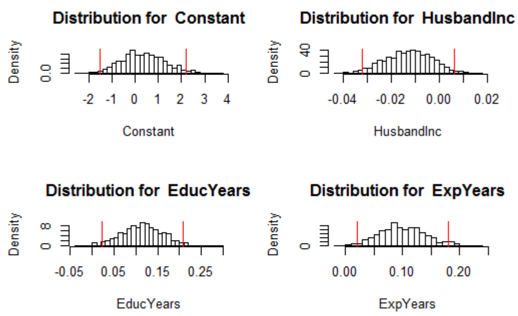

Figure 5: Distribution of the parameters with its 95% confidence interval (I)

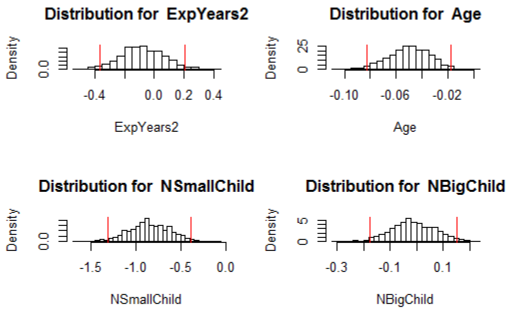

Figure 6: Distribution of the parameters with its 95% confidence interval (II)

```
> BetaFeat
                    Mean        upper       lower
Constant      0.34057957  2.180953926 -1.49979478
HusbandInc   -0.01299479  0.006076275 -0.03206585
EducYears     0.11455707  0.207338016  0.02177613
ExpYears      0.10056627  0.179791545  0.02134099
ExpYears2    -0.07799728  0.207495176 -0.36348974
Age          -0.04989447 -0.017475928 -0.08231301
NSmallChild  -0.84840269 -0.388974933 -1.30783046
NBigChild    -0.01189881  0.152902525 -0.17670015
```

Now we have been asked to do a normal approximation which results can be seen here below. It can be seen that results are somehow similar, though not exact.

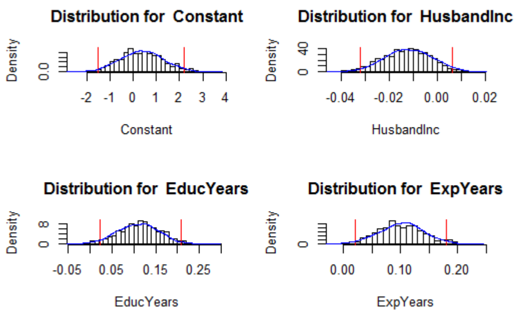

Figure 7: Distribution of the parameters with its 95% confidence interval and blue line for the normal approximation (I)

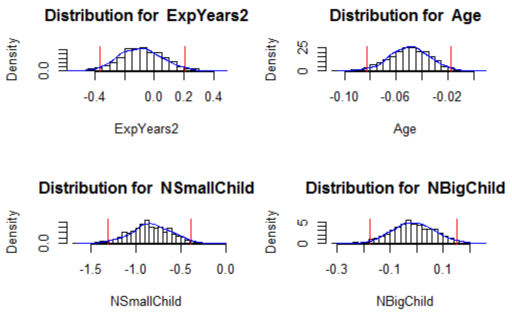

Figure 8: Distribution of the parameters with its 95% confidence interval and blue line for the normal approximation (II)

Also a table with the comparison of the different coefficients for both methods is reported.

```
 > ConclTAble
               GibbsMean     Gibbssd OptimalNBeta    Nsd
Constant      0.34057957 0.938966506  0.38451729 0.914146031
HusbandInc   -0.01299479 0.009730134 -0.01208712 0.009474897
EducYears     0.11455707 0.047337216  0.10840175 0.047218262
ExpYears      0.10056627 0.040421059  0.10221471 0.039459840
ExpYears2    -0.07799728 0.145659416 -0.09024608 0.140976385
Age          -0.04989447 0.016540073 -0.04962341 0.015890547
NSmallChild -0.84840269 0.234401919 -0.81567500 0.226635007
NBigChild    -0.01189881 0.084082316 -0.01241943 0.081506596
```

Contributions

All results and comments presented have been developed and discussed together by the members of the group.

Appendix

Question 1

```r

library(geoR)
library(mvtnorm)

set.seed(12345)

data<- read.csv("C:/Users/Carles/Desktop/Bayesian learning/Part3/rainfall.dat.txt")


########1A

##Initial Data


nDraws<-1000
data<- unlist(data)
mu_0<- 0
v_0<- 1
sigma2_0<- 1
tau2_0<- 10


Normal_model<- function(nDraws, data,mu_0, v_0, tau2_0, sigma2_0){
  require(geoR)
  ##Initializing data result
  gibbsDraws <- matrix(0,nDraws,2)
  colnames(gibbsDraws)<-c("mu", "sigma2")

  #basic variables
  n<- length(data)
  v_n<- n +v_0

  #Initializing basic variables for the gibb sampling
  mu<-mu_0

  for(i in 1:nDraws){

    sigma2 <- rinvchisq(1, df= v_n, scale = (v_0*sigma2_0+ sum((data-mu)**2))/(n +v_0))
    gibbsDraws[i,2] <- sigma2

    w<- (n/sigma2)/(n/sigma2+1/tau2_0)
    mu_n<- w*mean(data)+(1-w)*mu_0
    invtau2<- (1/(n/sigma2+1/tau2_0))

    mu <-rnorm(1,mu_n, sd = sqrt(invtau2))
    gibbsDraws[i,1]<- mu

  }

  return(gibbsDraws)


}

gibbsDraws<-Normal_model(nDraws=nDraws, data= data ,mu_0= mu_0, v_0= v_0, tau2_0= tau2_0, sigma2_0= sigma2_0)
tail(gibbsDraws)

par(mfrow = c(1,3))
plot(gibbsDraws, type = "l", main = "posterior distribution of the data")
plot(gibbsDraws[,1], type = "l", ylab = "mu", main = "convergence of mu")
plot(gibbsDraws[,2], type = "l", ylab = "sigma", main = "convergence of sigma")



hist(gibbsDraws[,1], freq = FALSE, main='Gibbs draws', ylim = c(0,0.5), xlab= "mu")
lines(seq(-2,4,by=0.01),dnorm(seq(-2,4,by=0.01), mean = 1), col = "red", lwd = 3)
plot(cumsum(gibbsDraws[,1])/seq(1,nDraws),type="l", main='Gibbs draws', xlab='Iteration number', ylab='cumulative mean of mu')
lines(seq(1,nDraws),1*matrix(1,1,nDraws),col="red",lwd=3)
acf(gibbsDraws[,1], main='Gibbs draws', lag.max = 20)
par(mfrow = c(1,1))

###############1b

##########    BEGIN USER INPUT #################
# Data options
rawData <- data
x <- as.matrix(data)

# Model options
```

```
nComp <- 2    # Number of mixture components

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(0,nComp) # Prior mean of theta
tau2Prior <- rep(10,nComp) # Prior std theta
sigma2_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)
nu0 <- rep(4,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)
  thetaDraws <- matrix(NA,nCat,1)
  for (j in 1:nCat){
    thetaDraws[j] <- rgamma(1,param[j],1)
  }
  thetaDraws = thetaDraws/sum(thetaDraws) # Diving every column of ThetaDraws by the sum of the
       elements in that column.
  return(thetaDraws)
}

# 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 allocations.
theta <- 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))


##Recording mu and sigma2

matmu<- matrix(0, nrow = nIter, ncol = nComp)
colnames(matmu)<- c("mu1", "mu2")

matsigma2<- matrix(0, nrow = nIter, ncol = nComp)
colnames(matsigma2)<- c("sigma1", "sigma2")

matpi<- matrix(0, nrow = nIter, ncol = nComp)
colnames(matsigma2)<- c("pi1", "pi2")


for (k in 1:nIter){
  message(paste('Iteration number:',k))
  alloc <- S2alloc(S) # Just a function that converts between different representations of the
       group allocations
  nAlloc <- colSums(S)
  print(nAlloc)
  # Update components probabilities
  w <- rDirichlet(alpha + nAlloc)
  matpi[k,]<- w


  # Update theta'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
    theta[j] <- rnorm(1, mean = muPost, sd = sqrt(tau2Post))
  }
  matmu[k,]<- muPost


  # 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] - theta[j])^2))/(nu0[j] + nAlloc[j]))
    }
  matsigma2[k,]<- sigma2


  # Update allocation
  for (i in 1:nObs){
    for (j in 1:nComp){
      probObsInComp[j] <- w[j]*dnorm(x[i], mean = theta[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)){
    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,theta[j],sd = sqrt(sigma2[j]))
      mixDens <- mixDens + w[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)
  }

}

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)

####C Graphical representation
#Data sets for the distribution of mu1, mu2, sigma2_1, sigma_2, pi1, pi2
matmu
matpi
matsigma2
#Data set for the distribution of mu1,  sigma2_1 in the first case



sampled_simple<-rnorm(1000, mean = mean(gibbsDraws[,1]), sd = sqrt(mean(gibbsDraws[,2])))

mixture_sample<- mean(matpi[,1])*rnorm(1000, mean = mean(matmu[,1]), sd = sqrt(mean(matsigma2[,1])))+ (1-mean(matpi[,1]))*rnorm(1000, mean = mean(matmu[,2]), sd = sqrt(mean(matsigma2[,2])))

hist(data, freq = FALSE, main='Gibbs draws', xlab= "mu", breaks = 30)
lines(density(sampled_simple), col = "red")
lines(density(mixture_sample), col = "yellow")
legend("topright", box.lty = 1, legend = c("Data histogram","Normal density", "Mixture density"), col=c("black","red","yellow"), lwd = 2)
```

Question 2

```

########Question 2
```

```r
#install.packages("msm")
library(msm)

Data<- read.csv("C:/Users/Carles/Desktop/Bayesian learning/Part2/WomenWork.dat.txt", sep ="", header = TRUE)
glmModel <-   glm(Work ~ 0 + ., data = Data, family = binomial(link = "probit"))
summary(glmModel)

#### Variables
tau <- 10;          # Prior scaling factor such that Prior Covariance = (tau^2)*I
chooseCov <- c(1:8)  # Here we choose which covariates to include in the model


y <- as.vector(Data[,1]); # Data from the read.table function is a data frame. Let's convert y and X to vector and matrix.
X <- as.matrix(Data[,2:ncol(Data)]);
initVal <- as.vector(rep(0,dim(X)[2]));
covNames <- names(Data)[2:length(names(Data))];
X <- X[,chooseCov]; # Here we pick out the chosen covariates.
covNames <- covNames[chooseCov];
nPara <- dim(X)[2];



# Setting up the prior
mu_0    <- as.vector(rep(0,nPara)) # Prior mean vector
Omega_0   <- tau^2*diag(nPara);
beta_0   <- as.vector(rmvnorm(1, mean = mu_0, sigma =Omega_0))
nIter   <- 1000



ProbitFunction<- function(beta_0, mu_0, Omega_0, X, y, n_iter){
  posy_0  <- which(y == 0)
  posy_1  <- which(y == 1)
  nPara <- dim(X)[2]


  beta<-matrix(0, ncol = nPara, nrow = n_iter)
  beta[1,]<- beta_0
  B<- solve(solve(Omega_0)+t(X)%*%X)

  for(i in 1:n_iter){
    beta[i, ]<- rmvnorm(1, as.vector(B%*%(solve(Omega_0)%*%mu_0+t(X)%*%y)), sigma = B)

    y[posy_1] <- rtnorm(length(posy_1), mean = as.vector(X[posy_1,]%*%beta[i,]), sd = 1,lower = 0, upper = Inf)
    y[posy_0] <- rtnorm(length(posy_0), mean = as.vector(X[posy_0,]%*%beta[i,]), sd = 1,lower = -Inf, upper = 0)



    }
  return(beta)
}


Betadist<-ProbitFunction(beta_0=beta_0, mu_0= mu_0, Omega_0= Omega_0, X= X, y= y , n_iter = nIter)
colnames(Betadist)<- covNames
dim(Betadist)
BetaFeat<- data.frame(Mean = apply(Betadist, 2, mean),
                      upper = apply(Betadist, 2, mean)+1.96*apply(Betadist, 2, sd),
                      lower = apply(Betadist, 2, mean)-1.96*apply(Betadist, 2, sd))
par(mfrow = c(2,2))
for(i in 1: 4){
    hist(Betadist[,i], main = paste("Distribution for " ,covNames[i]), xlab = covNames[i], breaks = 30, freq = FALSE)
    abline(v = BetaFeat$lower[i], b = 0, col = "red")
    abline(v = BetaFeat$upper[i], b = 0, col = "red")

  }
par(mfrow = c(2,2))
for(i in 5:8){
  hist(Betadist[,i], main = paste("Distribution for " ,covNames[i]), xlab = covNames[i], breaks = 30, freq = FALSE)
  abline(v = BetaFeat$lower[i], b = 0, col = "red")
  abline(v = BetaFeat$upper[i], b = 0, col = "red")

}


###C

LogPostProbit <- function(betaVect,y,X,mu,Sigma){
  nPara <- length(betaVect);
  linPred <- X%*%betaVect;
```

```
  # The following is a more numerically stable evaluation of the log-likelihood in my slides:
  # logLik <- sum(y*log(pnorm(linPred)) + (1-y)*log(1-pnorm(linPred)) )
  logLik <- sum(y*pnorm(linPred, log.p = TRUE) + (1-y)*pnorm(linPred, log.p = TRUE, lower.tail
      = FALSE))

  # 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)

}

OptimResults<-optim(initVal,LogPostProbit,gr=NULL,y,X,mu= mu_0,Sigma= Omega_0,method=c("BFGS"),
    control=list(fnscale=-1),hessian=TRUE)

BetaCoef<- OptimResults$par
names(BetaCoef)<- covNames
J<--solve(OptimResults$hessian)

myconf95<-c(lowbound =BetaCoef[6]-1.96*sqrt(J[6,6]), highbound=BetaCoef[6]+1.96*sqrt(J[6,6]))

sampleBetaProbit<- matrix(ncol = length(BetaCoef), nrow = 1000)
for(i in 1:length(BetaCoef)){
sampleBetaProbit[,i]<- rnorm(1000, mean = BetaCoef[i], sd = sqrt(J[i,i]))
}

par(mfrow = c(2,2))
for(i in 1: 4){
  hist(Betadist[,i], main = paste("Distribution for " ,covNames[i]), xlab = covNames[i], breaks
       = 30, freq = FALSE)
  abline(v = BetaFeat$lower[i], b = 0, col = "red")
  abline(v = BetaFeat$upper[i], b = 0, col = "red")
  lines(density(sampleBetaProbit[,i]), col = "blue")
}
for(i in 5:8){
  hist(Betadist[,i], main = paste("Distribution for " ,covNames[i]), xlab = covNames[i], breaks
       = 30, freq = FALSE)
  abline(v = BetaFeat$lower[i], b = 0, col = "red")
  abline(v = BetaFeat$upper[i], b = 0, col = "red")
  lines(density(sampleBetaProbit[,i]), col = "blue")


}

ConclTAble<- data.frame(GibbsMean = apply(Betadist, 2, mean),
                        Gibbssd = apply(Betadist, 2, sd),
                        OptimalBeta = BetaCoef,
                        Gibbssd=sqrt(diag(J)))
```