# 732A91: Lab 1
# Bayesian Learning

Sarah Alsaadi, Carles Sans Fuentes

April 13, 2017

---

## Question 1

Let $y_1, \ldots, y_n|\theta \sim \text{Bern}(\theta)$, and assume that you have obtained a sample with $s = 14$ successes in $n = 20$ trials. Assume a $\text{Beta}(\alpha_0, \beta_0)$ prior for $\theta$ and let $\alpha_0 = \beta_0 = 2$.

- Draw random numbers from posterior $\theta|y_1, \ldots, y_n \sim \text{Bern}(\alpha_0 + s, \beta_0 + f)$ and verify graphically that the posterior mean and standard deviation converges to the true mean $E[\theta] = \frac{\alpha_0+s}{\alpha_0+s+\beta_0+f} \approx 0.66$ and true standard deviation $Var(\theta) = \frac{(\alpha_0+s)*(\beta_0+f)}{(\alpha_0+s+\beta_0+f)^2*(\alpha_0+s+\beta_0+f+1)} \approx 0.09$. Figure 1 and 2 shows how the mean and standard deviation of samples randomly drawn converges to the true mean and true standard deviation with larger samples.

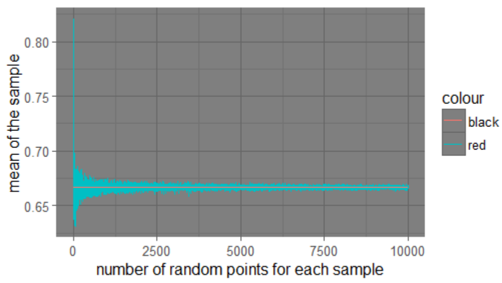


Figure 1: A plot representing the mean of 10000 samples where the sample size goes from 1 to 10000. The theoretical mean is plotted as a line.

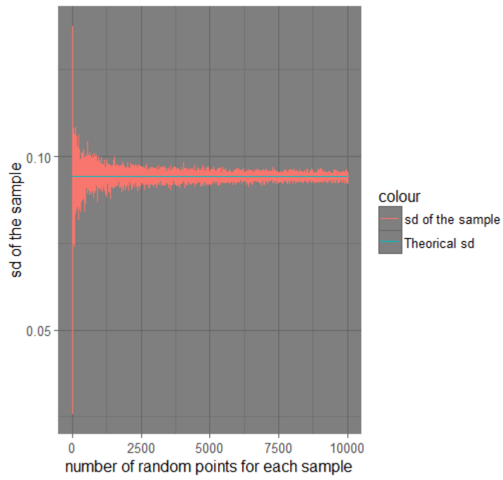


Figure 2: A plot representing the standard deviation of 10000 where the sample size goes from 1 to 10000. The theoretical standard deviation (sd) is plotted as a line.

- Use simulation (nDraws= 10000) to compute the posterior probability $Pr(\theta < 0.4|y)$ and compare with the exact value $Pr(\theta < 0.4|y) = 0.00397$. The simulated value is given below, we get 0.0036 which is close enough to the exact value.

```
pbeta(0.4,16,8)
[1] 0.003972563
simulated pbeta
[1] 0.0036
```

- Compute the posterior distribution of the log-odds $\phi = log(\frac{\theta}{1-\theta})$ by simulation with nDraws= 10000. Figure 3 below shows the histogram together with the kernel density of the data simulated from the posterior distribution of the log-odds $\phi = log(\frac{\theta}{1-\theta})$ with nDraws= 10000.

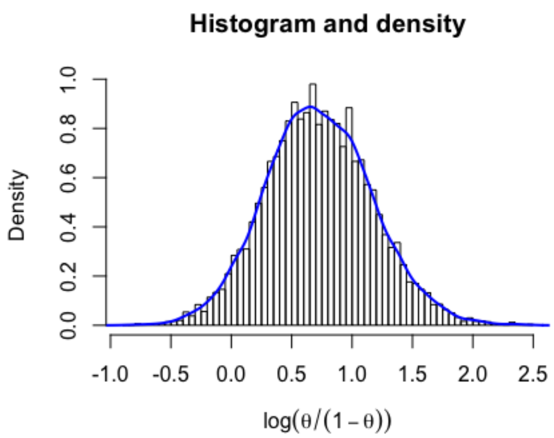


Figure 3: Histogram and kernel density of of the data simulated from the posterior distribution of the log-odds $\phi = log(\frac{\theta}{1-\theta})$ with nDraws= 10000.

## Question 2

Assume that you have asked 10 randomly selected persons about their monthly income (in thousands Swedish Krona) and obtained the following 10 observations $(y_1, \ldots, y_{10}) = (14, 25, 45, 25, 30, 33, 19, 50, 34, 67)$. Assume $y_1, \ldots, y_n|\mu, \sigma^2 \sim logNormal(\mu, \sigma^2)$,

$\mu = 3.5$ and a non-informative prior $p(\sigma^2) \propto \frac{1}{\sigma^2}$. It can be shown that the posterior for $\sigma^2$ is $Inv - \chi^2(n, \tau^2)$ (scaled), where

$$\tau^2 = \frac{\sum_{i=1}^{n}(log(y_i) - \mu)}{n}$$

.

- Simulate 10000 draws from the posterior of $\sigma^2$ and compare with the theoretical $Inv - \chi^2(n, \tau^2) = Inv - \chi^2(10, 0.198)$. Figure 4 shows the histogram of the data simulated from $Inv-\chi^2(10, 0.198)$ and the density of the theoretical posterior distribution $Inv-\chi^2(10, 0.198)$. Since the simulated sample is so big, the histogram and the density is quite alike.

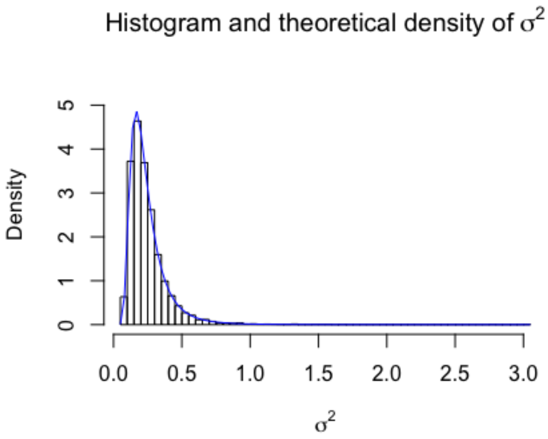


Figure 4: Histogram of simulated data and theoretical density of the $\sigma^2$ with draws= 10000.

- The most common measure of income inequality is the Gini coefficient, G, where $0 \leq G \leq 1$. G= 0 means a completely equal income distribution, whereas G= 1 means complete income inequality. It can be shown that $G = 2\Phi(\frac{\sigma}{\sqrt{2}}) - 1$ when incomes follow a $logNormal(\mu, \sigma^2)$ distribution. Use the draws in a) to compute the posterior of the Gini coefficient for the current data set. Figure 5 shows the histogram of G and the kernel density. It shows that G is far from 1 so the income distribution seems to be closer to equal than inequal.

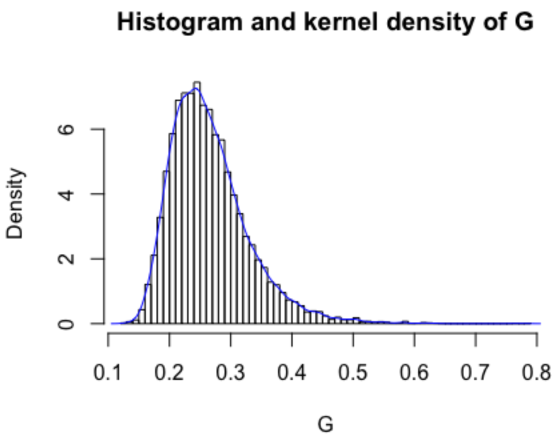


Figure 5: Histogram of simulated data and kernel density of G with draws= 10000.

- Use the posterior draws from b) to compute a 95% equal tail credible set for G. Also, do a kernel density estimate of the posterior of G and use it to compute a 95% Highest Posterior Density set for G. The Highest Posterior Density set (HPD) is a credible set that consists of

$\theta$-values having the Highest Posterior Density. Below we have a 95% equal tail credible set for G and and HPD. For a symmetric distribution, the intervals would be the same but since the distribution of G is slightly skewd, the intervals differ.

```
quantile(G,  probs = c(0.025, 0.975))
#     2.5%      97.5%
# 0.1739323 0.4152031
hdi(density(G),credMass=0.95)
#lower     upper
#0.1601422 0.3918670
```

# Question 3

The following data is the observed wind directions at a given location on 10 different days. The data recorded in radians:

$$(y_1, \ldots, y_{10}) = (-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)$$

where $-\pi \leq y \leq \pi$.
Assume that the observations are independent and follow the von Mises distribution

$$p(y|\mu, \kappa) = \frac{exp(\kappa * cos(y - \mu))}{2\pi I_0(\kappa)}$$

$-\pi \leq y \leq \pi$.
Furthermore, assume that $\mu = 2.39$ and let $\kappa \sim \text{Exp}(\lambda = 1)$ apriori.

- Plot the posterior distribution of $\kappa$ for the data given above. The posterior is given by the following:

$$p(\kappa|\mu, y) = exp(-\kappa) \prod_{i=1}^{10} \frac{exp(\kappa * cos(y_i - 2.39))}{2\pi I_0(\kappa)}$$

  Figure 6 shows the posterior of $\kappa$.

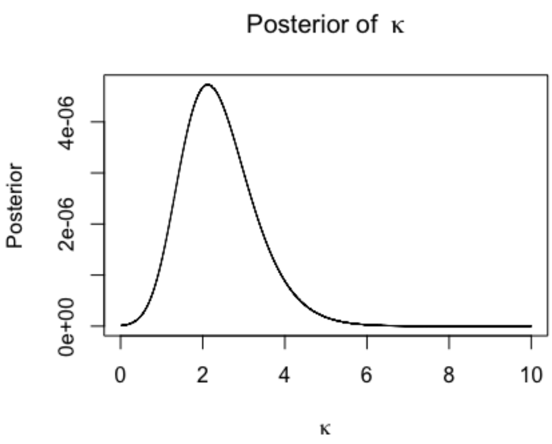


Figure 6: Posterior of $\kappa$ .

- The approximate point where the posterior has its mode is $(2.125, 4.7 * 10^{-6})$.

# Contributions

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

# Appendix

## Question 1

```
install.packages("geoR")
install.packages("HDInterval")
library("geoR")
library("HDInterval")
#lab 1 Bayesian learning

#1a
x<-rbeta(10000,2+14,2+6)
h<-hist(x,breaks = seq(0,1,0.02))
xfit<-seq(min(x),max(x),length=100)
yfit<-dbeta(xfit,16,8)
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)

#1b
pbeta(0.4,16,8)
y<-x[x<0.4]
prob<-length(y)/length(x)


#1c
xodds<-log(x/(1-x))
hist(xodds,breaks = 50,prob=TRUE, main = 'Histogram and density', xlab = expression(paste(log(theta/(1-theta)))))
lines(density(xodds), col="blue", lwd=2)

#2a

data<-c(14,25,45,25,30,33,19,50,34,67)

tao2<-function(data)
{
  sum((log(data)-3.5)^2)/length(data)

}

tao<-tao2(data=data)

sigma2<-rinvchisq(10000,df=10,scale=tao)
hist(sigma2,breaks = 100, prob=TRUE)
x<-seq(from=0, to=10000, by=0.001)
hx<-dinvchisq(x,df=10,scale=tao)

sigma2.histogram = hist(sigma2, breaks = 100, freq = F)
sigma2.ylim.normal = range(0, sigma2.histogram$density, dinvchisq(sigma2,df=10,scale=tao), na.rm = T)
hist(sigma2, breaks = 100, freq = F, ylim = c(0, 5.5), main=expression("Histogram and theoretical density of" ~ sigma^2), xlab = expression(paste(sigma^2)), ylab = 'Density')
curve(dinvchisq(x,df=10,scale=tao), add = T,col="blue")

#2.b
sigma<-sqrt(sigma2)/sqrt(2)
G<-2*pnorm(sigma,0,1)-1
hist(G,freq=F,breaks=50, main='Histogram and kernel density of G')
lines(density(G),col="blue")

#2.c
quantile(G,  probs = c(0.025, 0.975))
#    2.5%     97.5%
# 0.1739323 0.4152031

hdi(density(G),credMass=0.95)

#lower     upper
#0.1601422 0.3918670

#3.a

windradians<-c(-2.44,2.14,2.54,1.83,2.02,2.33,-2.79,2.23,2.07,2.02)
posterior<-function(k)
{
  return(exp(-k)*prod(exp(k*cos(windradians-2.39))/(2*pi*besselI(x=k,nu=0))))

}


values<-sapply(k,posterior)
plot(k,values,type="l",main="Posterior of " ~ kappa, xlab=expression(paste(kappa)), ylab="Posterior")
```

```
#3.b
max(values)
#4.727694e-06
k[which.max(value)]
#2.125
```

# 732A91: Lab 2
# Bayesian Learning

Sarah Alsaadi, Carles Sans Fuentes

May 22, 2017

---

## Linear and polynomial regression

In this exercise we use the dataset TempLinkoping.txt in our analysis. The dataset contains daily temperatures (in Celsius degrees) at Malmslatt and Linkoping over the course of the year 2016, having as a response variable temp and covariate time.

First we determine the prior distribution of the model parameter. Given that our likelihood is a quadratic regression (see the summary of the quadratic regression model below) and that its conjugate prior is a multivariate normal distribution (because there is more than one Beta parameter), the prior hyperparameters are chosen as following:

1. $\mu_0$ is the linear coefficients from our data (our best guess is just the computation of the parameters as if it was a quadratic regression of our data),
2. $\omega_0$ a diagonal matrix of 1s given that all data has the same importance,
3. $v_0$ equal 6 in order to not to give too much importance to our prior and
4. $\sigma_0^2$ equal 16 (similar variance of the quadratic regression model)

```
> summary(lm)

Call:
lm(formula = temp ~ time + I(time^2), data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-10.0408  -2.6971  -0.1414   2.5157  12.2085

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -10.6754     0.6475  -16.49   <2e-16 ***
time         93.5980     2.9822   31.39   <2e-16 ***
I(time^2)   -85.8311     2.8801  -29.80   <2e-16 ***
---

Residual standard error: 4.107 on 363 degrees of freedom
Multiple R-squared:  0.7318,  Adjusted R-squared:  0.7304
F-statistic: 495.3 on 2 and 363 DF,  p-value: < 2.2e-16
```

In order to check whether our prior is sensible, we have simulated 1000 draws from the joint prior of all parameters and for every draw compute the regression curve. In order to check whether our prior look properly, we have evaluated the distributions of the Beta parameters and comparing it with the one from the quadratic regression. Below we have the mean of the beta coefficients.

```
apply(betasprior, 2, mean)
     beta0      beta1      beta2
 -9.991929  92.984489 -85.085894
```

It can be seen that given that our prior is really similar to our data, the Beta mean coefficients are almost the same.

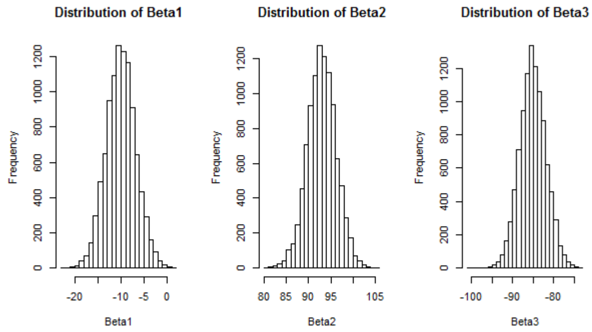


Figure 1: Histogram plot distribution of the our 3 Beta prior after 10000 simulations

It can be seen that the distribution looks normally distributed through the mean parameter of our data. This is good given the prior used. For this reason, we can say the curve looks reasonable.

Now, we have written a program that simulates from the joint posterior distribution of $\beta_0, \beta_1, \beta_2 and \sigma_2$. We have produced a scatter plot of the temperature data and overlay a curve for the posterior mean of the regression function $f(time) = \beta_0 + \beta_1 * time + \beta_2 * time^2$ as well as the 95% equal tail posterior probability intervals for every value of time and then connect the lower and upper limits of the interval by curves ( see figure 2)

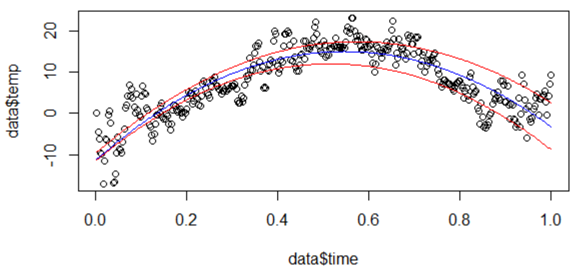


Figure 2: Histogram of the data with the posterior mean distribution of the data (Blue line) and its 95% credible interval (red lines)

It can be seen that the variance of the data increases for the last half of the prediction.

In 1d we are asked to locate the time with the highest expected temperature (that is, the time where E(temp—time) is maximal).We have used the previous betas to simulate the new data from the highest value of temp given time and we have plot the distribution of this points (see figure 3 below). It is seen that the mean of the maximum point is around 14-16. This makes sense since it is similar to the data we got.

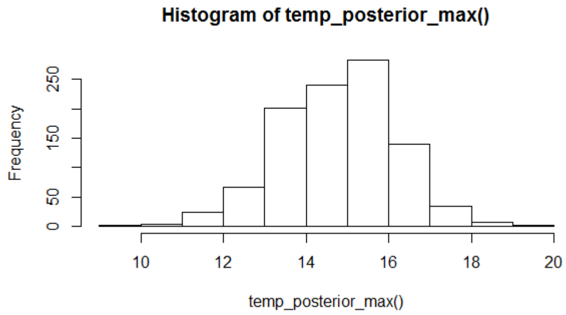


Figure 3: Scatterplot of the data together with the computed tempreature ( blue curve) from the posterior mean of $\beta_0, \beta_1, \beta_2$ and the lower 2.5% and the upper 97.5% posterior credible interval

Finally, we are asked to estimate a polynomial model of order 7 and we are told that higher order terms may not be needed. Since we have a strong belief that higher order terms are not needed, we specify the prior parameters for those coefficients to reflect that. This would be a an expected value close to 0 and a very small variance.

## Question 2

In activity 2 we are asked to write the model with the logistic function a data set to predict the probability that a woman will work given some variables that defines her.

For that, first we fit our data to the logistic model directly, and we get the following:

```
> summary(glmModel)

Call:
glm(formula = Work ~ 0 + ., family = binomial, data = Womenwork)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1662  -0.9299   0.4391   0.9494   2.0582

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
Constant     0.64430    1.52307   0.423 0.672274
HusbandInc  -0.01977    0.01590  -1.243 0.213752
EducYears    0.17988    0.07914   2.273 0.023024 *
ExpYears     0.16751    0.06600   2.538 0.011144 *
ExpYears2   -0.14436    0.23585  -0.612 0.540489
Age         -0.08234    0.02699  -3.050 0.002285 **
NSmallChild -1.36250    0.38996  -3.494 0.000476 ***
NBigChild   -0.02543    0.14172  -0.179 0.857592
---

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 277.26  on 200  degrees of freedom
Residual deviance: 222.73  on 192  degrees of freedom
AIC: 238.73

Number of Fisher Scoring iterations: 4
```

Now we are asked to approximate the posterior distribution of the 8-dim parameter vector $\beta$ with a multivariate normal distribution. For that, we have implemented a logistic function and used the optim() from R to find the Hessian and the best parameters for the best case given optimizing the posterior. The results is the following one:

```
> OptimResults
$par
[1] -0.020734822  0.198537870  0.170970267 -0.157370056 -0.073756571 -1.308417158
[7]  0.002373846

$value
[1] -134.015
```

```

$counts
function gradient
      59       14

$convergence
[1] 0

$message
NULL

$hessian
             [,1]        [,2]         [,3]         [,4]        [,5]        [,6]
[1,] -21691.9214 -10335.9656  -7724.06469 -1095.904557 -33606.6413 -190.535283
[2,] -10335.9656  -6069.9347  -4546.30708  -647.404832 -19675.4015 -127.202981
[3,]  -7724.0647  -4546.3071  -5353.80184  -982.743357 -16236.0124  -75.200938
[4,]  -1095.9046   -647.4048   -982.74336  -214.881014  -2470.3813   -7.716988
[5,] -33606.6413 -19675.4015 -16236.01237 -2470.381294 -68796.8982 -321.680446
[6,]   -190.5353   -127.2030    -75.20094    -7.716988   -321.6804  -11.942247
[7,]   -996.0942   -584.7677   -373.77395   -42.860366  -1894.1299  -12.925833
             [,7]
[1,]  -996.09418
[2,]  -584.76769
[3,]  -373.77395
[4,]   -42.86037
[5,] -1894.12992
[6,]   -12.92583
[7,]  -123.33156
```

Also, the 95% CI for the variable NSmallChild must be found (see below) in order to define whether the variable is important or not

```
lowbound highbound
-2.028515 -0.588319
```

It can be seen that the credible interval is far from 0, which means that the variable is an important feature.

Finally, we are asked to Write a function that simulates from the predictive distribution of the response variable in a logistic regression.We have used our previously normal approximation to simulate and plot the predictive distribution for the Work variable for a 40 year old women, with two children (3 and 9 years old), 8 years of education, 10 years of experience. and a husband with an income of 10. The results is the following histogram:

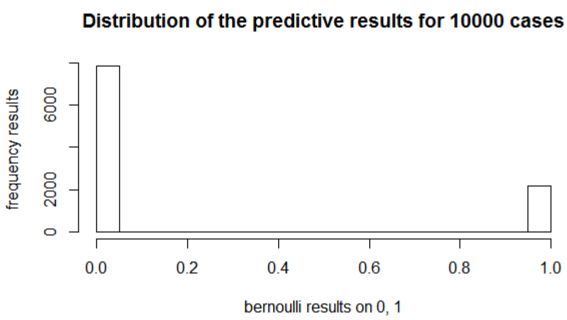


Figure 4: Histogram of the predictive distribution of the response variable in a logistic regression

The result shows that it is quite much probable that the Woman will not be able to work.

# Contributions

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

# Appendix

## Question 1

```r
#lab 2 Bayesian learning

# install.packages("mvtnorm")
# install.packages("MASS")
library("mvtnorm")
library("geoR")
library("MASS")

data<-read.table("~/Google Drive/Kurser/Bayesian learning/Lab2/TempLinkoping2016.txt",head=TRUE)
data<- read.csv("C:/Users/Carles/Desktop/Bayesian learning/Part2/TempLinkoping2016.txt", sep ="\t")

#1.a
lm<-lm(temp~time+I(time^2),data)
summary(lm)

#1.b
mu_0<-c(-10,93,-85)
sigma_0<-16
v_0<-6
omega_0<-diag(3)
sigma_prior<-rinvchisq(1,df=v_0,scale=sigma_0)
beta_prior<-rmvnorm(n=1, mean =mu_0, sigma = sigma_prior*ginv(omega_0))
regress_prior<-beta_prior[1]+beta_prior[2]*data$time+beta_prior[3]*(data$time)^2
plot(y=data$temp,x=data$time)
lines(y=regress_prior,x=data$time,col="blue")

betasprior<-function(n)
{
  beta0<-numeric(n)
  beta1<-numeric(n)
  beta2<-numeric(n)

  for (i in 1:n)
  {
    beta0[i]<-as.vector(rmvnorm(n=1, mean =mu_0, sigma = sigma_prior*ginv(omega_0)))[1]
    beta1[i]<-as.vector(rmvnorm(n=1, mean =mu_0, sigma = sigma_prior*ginv(omega_0)))[2]
    beta2[i]<-as.vector(rmvnorm(n=1, mean =mu_0, sigma = sigma_prior*ginv(omega_0)))[3]

     }
  return(data.frame(beta0, beta1, beta2))
}
betasprior<-betasprior(10000)
apply(betasprior, 2, mean)
dim(betasprior)
par(mfrow=c(1,3))

hist(betasprior[,1], breaks = 30, xlab = "Beta1", main = "Distribution of Beta1")
hist(betasprior[,2], breaks = 30, xlab = "Beta2", main = "Distribution of Beta2")
hist(betasprior[,3], breaks = 30, xlab = "Beta3", main = "Distribution of Beta3")



#1.c
X<-cbind(rep(1,nrow(data)),data$time,(data$time)^2)
#beta_hat<-ginv(t(X)%*%X)%*%t(X)%*%data$temp
mu_n<-ginv(t(X)%*%X+omega_0)%*%(t(X)%*%data$temp+omega_0%*%mu_0)
Sigma_n<-t(X)%*%X+omega_0
v_n<-v_0+nrow(data)
sigma_n<-(1/v_n)*(v_0*sigma_0+(t(data$temp)%*%data$temp+t(mu_0)%*%omega_0%*%mu_0-t(mu_n)%*%Sigma_n%*%mu_n))

sigma_posterior<-as.vector(rinvchisq(1,df=v_n,scale=sigma_n))
beta_posterior<-rmvnorm(n=1, mean =mu_n, sigma = sigma_posterior*ginv(Sigma_n))
regress_posterior<-beta_posterior[1]+beta_posterior[2]*data$time+beta_posterior[3]*(data$time)^2
plot(y=data$temp,x=data$time)
lines(y=regress_posterior,x=data$time,col="blue")

betas<-function(n)
{
  beta0<-numeric(n)
  beta1<-numeric(n)
  beta2<-numeric(n)

  for (i in 1:n)
  {
    beta0[i]<-as.vector(rmvnorm(n=1, mean =mu_n, sigma = sigma_posterior*ginv(Sigma_n)))[1]
    beta1[i]<-as.vector(rmvnorm(n=1, mean =mu_n, sigma = sigma_posterior*ginv(Sigma_n)))[2]
```

```r
    beta2[i]<-as.vector(rmvnorm(n=1, mean =mu_n, sigma = sigma_posterior*ginv(Sigma_n)))[3]

    }
  return(data.frame(beta0,beta1, beta2))
  }


betas<-betas(1000)
betas0<-betas[,1]
betas1<-betas[,2]
betas2<-betas[,3]


temp_posterior<-function()
{
  n=1000
  k=366
  temp_posterior= matrix(data=NA, nrow=n, ncol=k)
  for(j in 1:k){
    for(i in 1:n){
      temp_posterior[i,j] = betas0[i]+betas1[i]*data$time[j]+betas2[i]*(data$time[j])^2
    }
  }
  return(temp_posterior)
  }
temp_posterior<-temp_posterior()

quantile_f<-function()
{
  k=366
  q_lower<-numeric(k)
  q_upper<-numeric(k)

  for (i in 1:k)
  {
    q_lower[i]=quantile(temp_posterior[,i],  probs = c(0.025, 0.975))[1]
    q_upper[i]=quantile(temp_posterior[,i],  probs = c(0.025, 0.975))[2]

    }
  return(data.frame(q_lower,q_upper))
}

quantile_f<-quantile_f()
plot(q_lower,type="l")


par(mfrow= c(1,1))
plot(y=data$temp,x=data$time)
lines( y =quantile_f[,2],x=data$time, col= "red")
lines(y =quantile_f[,1],x=data$time, col = "red")
lines(y=regress_posterior,x=data$time,col="blue")


myfunc<- function (data){
  regress_posterior<-betas[1]+betas[2]*data+betas[3]*(data)^2
  return(regress_posterior)

}

temp_mean<-mean(beta_posterior[,1])+mean(beta_posterior[,2])*data$time+mean(beta_posterior[,3])
    *data$time^2
t<-c(timeMax=data$time[which.max(temp_mean)], tempMax=max(temp_mean))


temp_posterior_max<-function()
{
  n=1000

  temp_posterior_max= numeric(n)

    for(i in 1:n){
      temp_posterior_max[i] = betas[i,1]+betas[i,2]*t[1]+betas[i,3]*t[1]^2
    }

  return(temp_posterior_max)
}

hist(temp_posterior_max())

##1d

lm7<-lm(temp~time+I(time^2)+I(time^3)+I(time^4)+I(time^5)+I(time^6)+I(time^7),data)
mu7<-lm7$coefficients
sigma7<-16
v7<-365
#First values
lambdaplot<-function(lambda){
```

```r
  lambda <-0.5
  Sigma7<-diag(8)*lambda
  sigma_prior7<-rinvchisq(1,df=v7,scale=sigma7)
  regress_prior7<- matrix(nrow = 1000, ncol = 366)
  for(i in 1:1000){
  beta_prior7<-rmvnorm(n=1, mean =mu7, sigma = sigma_prior7*ginv(Sigma7))
  regress_prior7[i,]<-beta_prior7[1]+beta_prior7[2]*data$time+beta_prior7[3]*(data$time)^2+beta
      _prior7[4]*(data$time)^3+beta_prior7[5]*(data$time)^4+beta_prior7[6]*(data$time)^5+beta_
      prior7[7]*(data$time)^6+beta_prior7[8]*(data$time)^7

  }
  return(colMeans(regress_prior7))
}
lambda05<-lambdaplot(lambda =0.5)
lambda1<-lambdaplot(lambda =1)
lambda5<-lambdaplot(lambda =10)
lambda10<-lambdaplot(lambda =100)

#lambda<-seq(from=0,to=10,by=1)
par(mfrow = c(2,2))
plot(y=data$temp,x=data$time, main = "lambda = 0.5")
lines(y=lambda05,x=data$time,col="red")
plot(y=data$temp,x=data$time, main = "lambda = 1")
lines(y=lambda1,x=data$time,col="red")
plot(y=data$temp,x=data$time, main = "lambda = 10")
lines(y=lambda5,x=data$time,col="red")
plot(y=data$temp,x=data$time, main = "lambda = 100 ")
lines(y=lambda10,x=data$time,col="red")
```

## Question 2

```r

#lab 2 Bayesian learning

#####################################################################
####2a
Womenwork<-read.table("~/Google Drive/Kurser/Bayesian learning/Lab2/WomenWork.dat.txt",head=
    TRUE)
Womenwork<- read.table("C:/Users/Carles/Desktop/Bayesian learning/Part2/WomenWork.dat.txt",head
    =TRUE)


glmModel <-   glm(Work ~ 0 + ., data = Womenwork, family = binomial)

summary(glmModel)

######b
#install.packages("mvtnorm") # Loading a package that contains the multivariate normal pdf
library("mvtnorm") # This command reads the mvtnorm package into R's memory. NOW we can use
    dmvnorm function.

# Loading data from file
#Data<-read.csv("/home/carsa564/Desktop/Bayesian learning/WomenWork.dat.txt", sep = "")
Data<- read.csv("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 2/Bayesian
    Learning/lab2/WomenWork.dat.txt", sep = "")
Data<-read.csv("~/Google Drive/Kurser/Bayesian learning/Lab2/WomenWork.dat.txt",sep ="")
Data<- read.csv("C:/Users/Carles/Desktop/Bayesian learning/Part2/WomenWork.dat.txt",sep ="")

tau <- 10;          # Prior scaling factor such that Prior Covariance = (tau^2)*I
chooseCov <- c(2:8)  # 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)]);
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 <- as.vector(rep(0,nPara)) # Prior mean vector
Sigma <- tau^2*diag(nPara);




LogPostLogistic <- function(betaVect,y,X,mu,Sigma){

  nPara <- length(betaVect);
  linPred <- X%*%betaVect;

  # evaluating the log-likelihood
  logLik <- sum( linPred*y -log(1 + exp(linPred)));
```

```
  if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite, stear 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)
}

initVal <- as.vector(rep(0,dim(X)[2]));

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

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

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



######2c

##J as the matrix of covariates for sigma is calculated before as well as BetaCoef which is mu
    or the BetaCoefficients
Woman<-c(10, 8, 10, 100, 40, 2,0)




logProv<- function(n, betas= BetaCoef, covariances= J,obs=Woman){
  prob<- integer(n)
  Pr<- integer(n)
  for(i in 1:n){
    myrandombetas<- as.vector(rmvnorm(1, mean= BetaCoef, sigma = covariances))
    Pr[i]<- exp((obs)%*%myrandombetas)/(1 + exp((obs)%*%myrandombetas))
    prob[i]<- rbinom(1,1,Pr[i])
  }

  return(list("Predictive results"= prob, "Probability results"= Pr))
}


distribution_of_the_predictive_results<- logProv(10000,  betas= BetaCoef, covariances= J,obs=
    Woman)

hist(distribution_of_the_predictive_results[[1]],
     xlab = "bernoulli results on 0, 1",
     ylab = "frequency results",
     main = "Distribution of the predictive results for 10000 cases")
```

# 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 $\mu$ and $\sigma^2$ 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 $\tau$, which has been chosen as 10. Proving with different $\tau$ values showed us that small $\tau$, 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 $\tau$ our optimal posterior $\mu$ distribution was lower (at around 26-28).

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

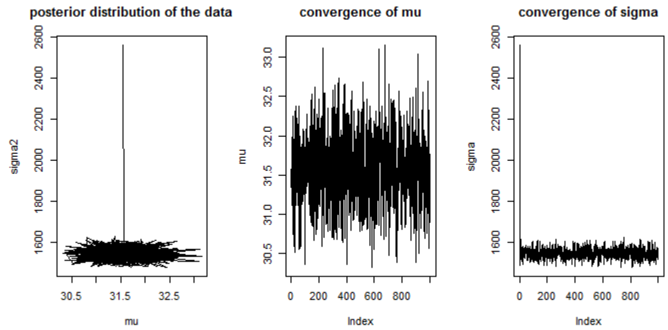


Figure 1: 3 plots of the posterior distribution of $\mu and \sigma$ with 1000 draws

In 2, we can see how the $\mu$ 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.

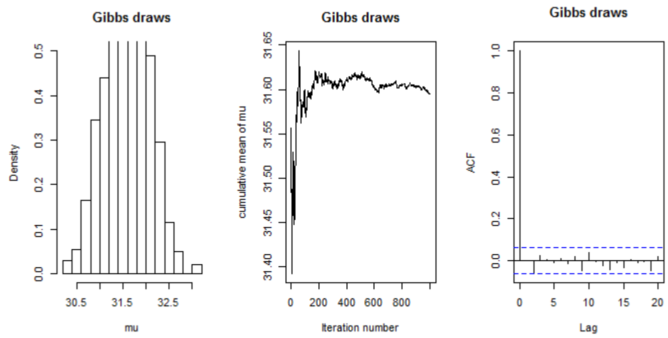


Figure 2: distribution of $\mu$ with 1000 draws,cumulative mean of the mean of $\mu$ and correlation lags

(c) In 1b, we are asked to run a mixture of normal on the data for 2 $\mu$ and $\sigma$, 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.

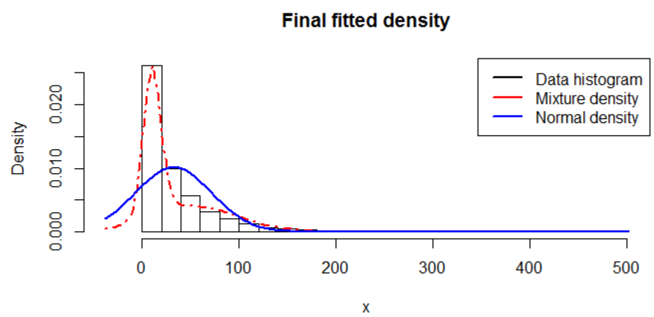


Figure 3: Convergence of the mixture density model after 100 iterations

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

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

.

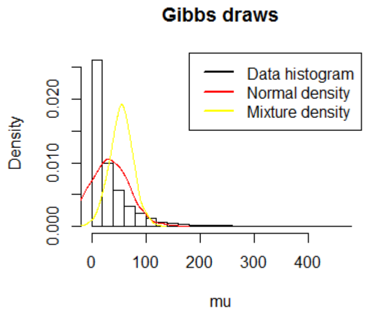


Figure 4: a histogram or kernel density estimate of the data, 2) Normal density $N(\mu, \sigma^2)$ in (a) and 3) Mixture of normals in(b)

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 $\beta$ 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 $\beta$ 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:

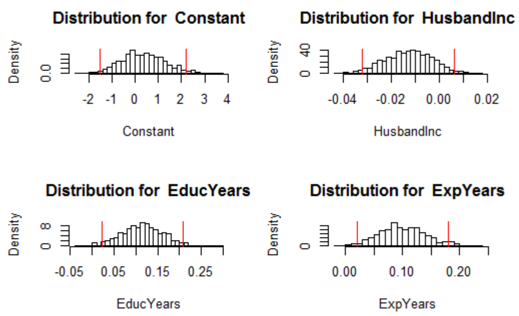


Figure 5: Distribution of the parameters with its 95% confidence interval (I)

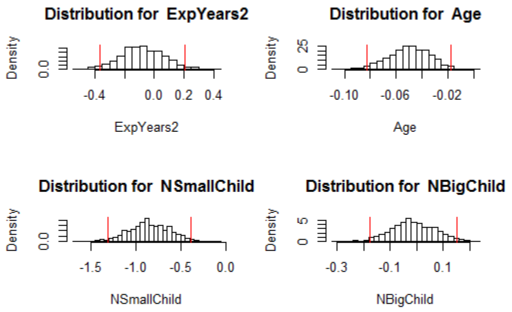


Figure 6: Distribution of the parameters with its 95% confidence interval (II)

```
> 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.

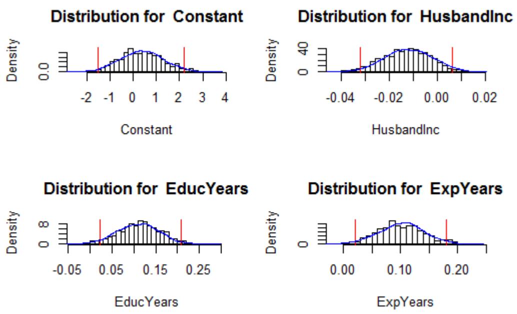


Figure 7: Distribution of the parameters with its 95% confidence interval and blue line for the normal approximation (I)

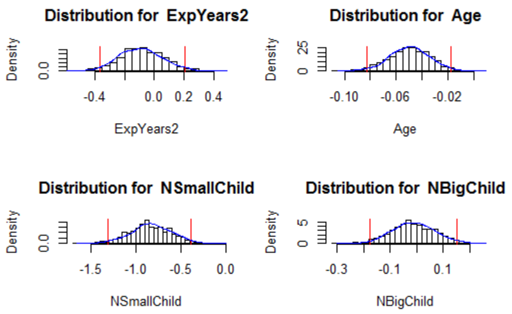


Figure 8: Distribution of the parameters with its 95% confidence interval and blue line for the normal approximation (II)

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

```

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)))
```

# 732A91: Lab 4
# Bayesian Learning

Sarah Alsaadi, Carles Sans Fuentes

May 24, 2017

---

## Poisson regression-the MCMC way

Consider the following Poisson regression model

$$y_i|\beta \sim \text{Poisson}[\exp(\mathbf{x}_i^T\beta)], i = 1, \ldots, n,$$

where $y_i$ is the count for the $i$th observation in the sample and $\mathbf{x}_i$ is the p-dimensional vector with covariate observations for the $i$th observation. The data set **eBayNumberOfBidderData.dat** 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 ($\mathbf{x}$):

- **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 opened 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) Using **glm** in R, we obtain the following results:

```
Call:
glm(formula = nBids ~ 0 + ., family = poisson, data = ebay)

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|)
Const        1.07244    0.03077  34.848  < 2e-16 ***
PowerSeller -0.02054    0.03678  -0.558   0.5765
VerifyID    -0.39452    0.09243  -4.268 1.97e-05 ***
Sealed       0.44384    0.05056   8.778  < 2e-16 ***
Minblem     -0.05220    0.06020  -0.867   0.3859
MajBlem     -0.22087    0.09144  -2.416   0.0157 *
LargNeg      0.07067    0.05633   1.255   0.2096
LogBook     -0.12068    0.02896  -4.166 3.09e-05 ***
MinBidShare -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: 6264.01  on 1000  degrees of freedom
```

```
Residual deviance:  867.47  on  991  degrees of freedom
AIC: 3610.3

Number of Fisher Scoring iterations: 5
```

The output shows that the significant MLE of the $\beta$:s are those for the covariates Const, VeryfyID, Sealed, MayBlem, LogBook and MinBidShare.

(b) In this exercise we did a Bayesian analysis of the Poisson regression. We let $\beta \sim N[0, 100(\mathbf{X}^T\mathbf{X})^{-1}]$ apriori, where $\mathbf{X}$ is the $n \times p$ covariate matrix. Next we assumed that the posterior density is approximately multivariate normal:

$$\beta|\mathbf{y}, \mathbf{X} \sim N(\tilde{\beta}, J_{\mathbf{y}}^{-1}(\tilde{\beta}))$$

where $\tilde{\beta}$ is the posterior mode and $J_{\mathbf{y}}(\tilde{\beta}) = -\frac{\partial^2 lnp(\beta|\mathbf{y})}{\partial\beta\partial\beta^T}|_{\beta=\tilde{\beta}}$ is the observed Hessian evaluated at the posterior mode. To obtain $\tilde{\beta}$ and $J_{\mathbf{y}}^{-1}(\tilde{\beta})$ we used numerical optimization (**optim.R**). The results of the estimated $\beta$:s are given in the table below. We also made histograms, one for each variable, of 10000 draws of $\beta$, the marginal distribution looks normal, see Figure 1.

| Variable | $\tilde{\beta}$ |
|---|---|
| **Const** | 1.06984118 |
| **PowerSeller** | −0.02051246 |
| **VerifyID** | −0.39300599 |
| **Sealed** | 0.44355549 |
| **MinBlem** | −0.05246627 |
| **MajBlem** | 0.22123840 |
| **LargNeg** | 0.07069683 |
| **LogBook** | −0.12021767 |
| **MinBidShare** | −1.89198501 |

Table 1: The obtained $\beta$:s through maximization of the posterior distirbution, one for each variable.

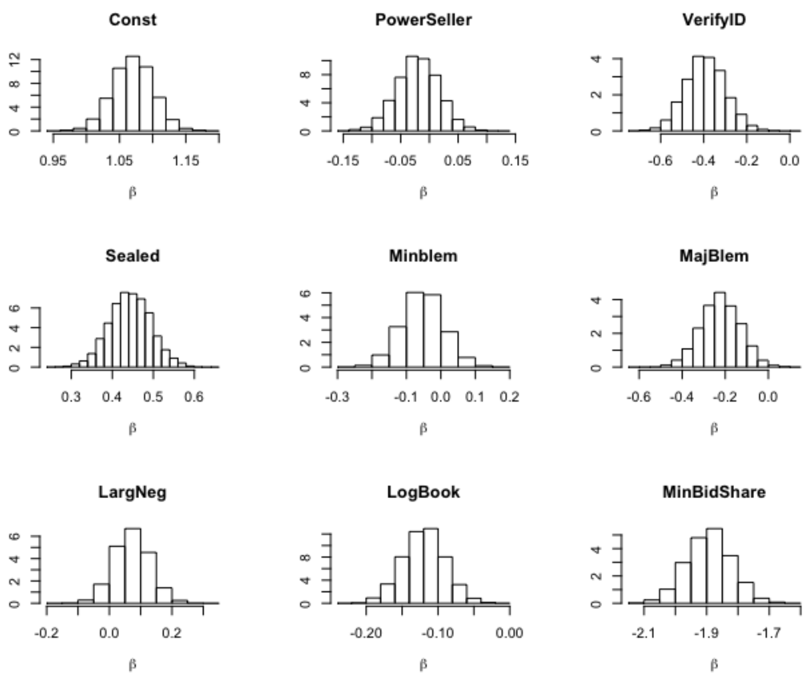


Figure 1: Histogram of the 10000 $\beta$:s drawn from the marginal posterior distribution

(c) In this exercise we simulate from the actual posterior of $\beta$ using the Metropolis algorithm and compare with the approximate results in b). We program a general function that uses the Metropolis algorithm to generate random draws from an arbitrary posterior density. We use the multivariate normal density as proposal density:

$$\theta_p|\theta_c \sim N(\theta_c, \tilde{c} \cdot \Sigma)$$

where $\Sigma = J_{\mathbf{y}}^{-1}(\tilde{\beta})$ obtained in the previous exercise and $\theta_c$ is the current draw (hence the subscript c). The value $\tilde{c}$ is a tuning parameter and is an input to our Metropolis function (so that a user can change it). The user of our Metropolis function is able to supply her own posterior density function, not necessarily for the Poisson regression, and still be able to use our Metropolis function.

First, one of the input arguments of our Metropolis function is called logPostFunc. logPostFunc is a function object that computes the log posterior density at any value of the parameter vector. This is needed when we compute the acceptance probability of the Metropolis algorithm. We program the log posterior density, since logs are more stable and avoids problems with too small or large numbers (overflow). Note that the ratio of posterior densities in the Metropolis acceptance probability can be written

$$\frac{p(\theta_p|\mathbf{y})}{p(\theta_c|\mathbf{y})} = \exp[log(p(\theta_p|\mathbf{y})) - log(p(\theta_c|\mathbf{y}))]$$

This is smart since the large or small common factors in $p(\theta_p|\mathbf{y})$ and $p(\theta_c|\mathbf{y})$ cancel out before we evaluate the exponential function (which can otherwise overflow).

Second, the first argument of our (log) posterior function is theta, the (vector) of parameters for which the posterior density is evaluated.

Third, the user's posterior density is also a function of the data and prior hyperparameters and those are supplied to the Metropolis function where we use the triple dot (...) argument which is like a wildcard for any parameters supplied by the user. This makes it possible to use the Metropolis function for any problem, even when a programmer don't know what the user's posterior density function looks like or what kind of data and hyperparameters being used in that particular problem.

Now, we use Metropolis function to sample from the posterior of $\beta$ in the Poisson regression for the eBay dataset. We assess MCMC convergence by graphical methods. The parameters $\phi_j = \exp(\beta_j)$ are usually considered more interpretable than the $\beta_j$. We compute the posterior distribution of $\phi_j$ for all variables.

Figure 2 shows that the Metropolis algorithm seem to converge, with some burn in period for all the $\beta$:s.

In Table 2 we have the mean posterior of the simulated $\beta$:s which are very similar to the ones obtained through maximization.

Figure 3 shows the marginal posterior for each $e^{\beta}$ obtained by the Metropolis algorit. The distribution looks normal.

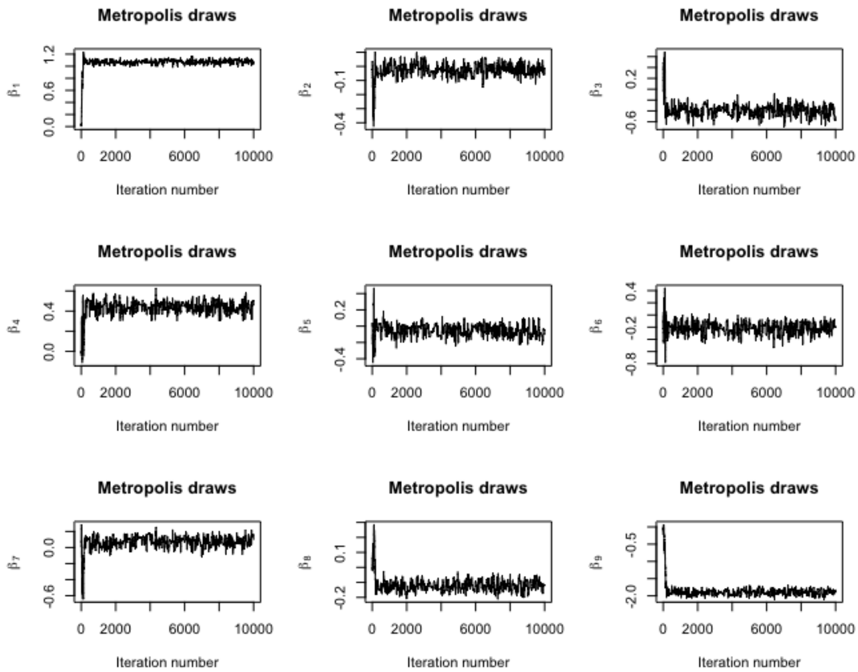


Figure 2: Plot of the 10000 $\beta$:s drawn from the marginal posterior distribution obtained by the Metropolis algorithm.

| Variable | $\tilde{\beta}$ |
|---|---|
| **Const** | 1.06538266 |
| **PowerSeller** | $-0.02529706$ |
| **VerifyID** | $-0.38552635$ |
| **Sealed** | 0.43553156 |
| **MinBlem** | $-0.05410433$ |
| **MajBlem** | $-0.22019352$ |
| **LargNeg** | 0.06493763 |
| **LogBook** | $-0.12021880$ |
| **MinBidShare** | $-1.88132053$ |

Table 2: The posterior mean of the simulated $\beta$:s, simulated using the Metropolis algorithm, the values are almost identical as the ones obtained through maximization

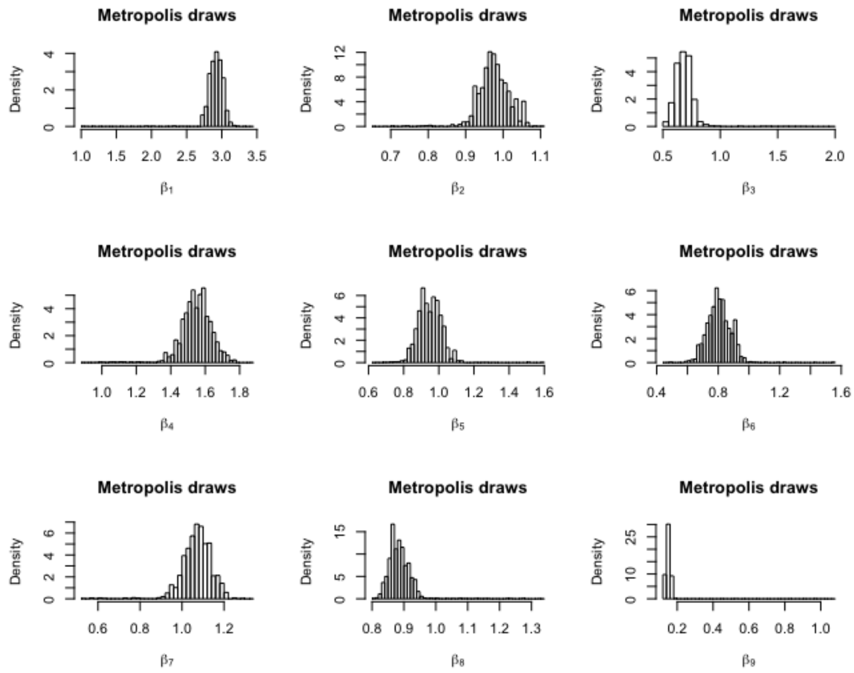


Figure 3: Histogram of the 10000 $e^{\beta}$ drawn from the marginal posterior distribution obtained by the Metropolis algorithm.

(d) We 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. The histogram of the predictive distribution is shown in Figure 4. The probability of no bidders in this new auction is 0.3512.

- **PowerSeller**= 1
- **VerifyID**= 1
- **Sealed**= 1
- **MinBlem**= 0
- **MajBlem**= 0
- **LargNeg**= 0
- **LogBook**= 1
- **MinBidShare**= 0.5

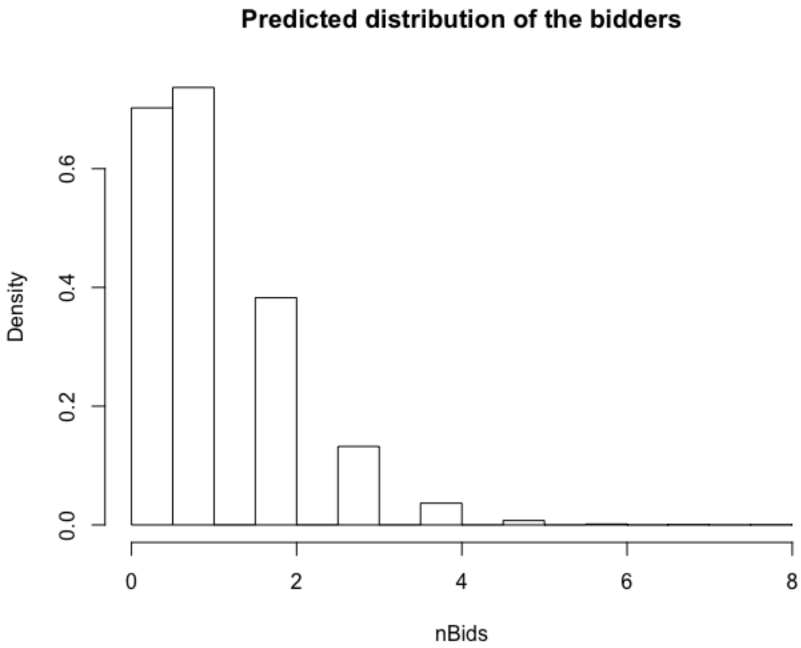


Figure 4: Histogram of 10000 simulated predicted values of the variable nBid with given characteristics

# Contributions

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

# Appendix

## Poisson regression-the MCMC way

```
### LAB 4
#1
Data<- read.csv("C:/Users/Carles/Desktop/Bayesian learning/Part4/eBayNumberOfBidderData.dat.txt", sep = "")
##a
#### Variables
PoisModel<-glm(nBids~.-Const, family="poisson", data = Data)
logLik(PoisModel)
mysum<-summary(PoisModel)[["coefficients"]]
##Significants at a 95% level
significant <- mysum[which(mysum[,4]<0.05),]
non_significant <- mysum[which(mysum[,4]>=0.05),]
list(significant = significant, nonsignificant = non_significant)
##b
chooseCov <- 1:(length(names(Data))-1);  # 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:length(names(Data))]);
covNames <- names(Data)[2:length(names(Data))];
X <- X[,chooseCov]; # Here we pick out the chosen covariates.
covNames <- covNames[chooseCov];
nPara <- dim(X)[2]
##prior Beta
library(geoR)
library(mvtnorm)
mu_0    <- matrix(0, nPara,1)
Sigma_0 <- 100*solve(crossprod(X,X))
Beta_0  <- rmvnorm(1, mean = mu_0, sigma = Sigma_0)
InitVal <- matrix(0, ncol=nPara, nrow =1)
LogPostPois <- function(betaVect,y = y,X = X, mu =mu_0,Sigma= Sigma_0){
  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(linPred*y-exp(linPred))
  # evaluating the prior
  logPrior <- dmvnorm(betaVect, mu, Sigma, log=TRUE);
  # add the log prior and log-likelihood together to get log posterior
  return(logLik + logPrior)
}
OptimResults<-optim(InitVal,LogPostPois,gr=NULL,y = y,X = X, mu =mu_0,Sigma= Sigma_0 ,method=c("BFGS"),control=list(fnscale=-1),hessian=TRUE)
BetaCoef<- OptimResults$par
colnames(BetaCoef)<- covNames
J<--solve(OptimResults$hessian)
mycoefvar<- data.frame(Coefficients= as.vector(BetaCoef), variance = diag(J))
rownames(mycoefvar)<- covNames
mycoefvar
######C
set.seed(12345)
LogPostPoisson <- function(theta, priormu, priorsigma, X, Y, ...) {
```

```r
  require(mvtnorm)
  likelihood <- dpois(Y, lambda = as.vector(exp((X) %*% t(theta))), log = TRUE)
  prior <- dmvnorm(theta, mean = priormu, sigma = priorsigma, log=TRUE)
  return(sum(likelihood) + prior)
}


metropl<-function(logPostFunc, theta_0, constant, sigma, nIter,...){
  #initialize chain
  require(mvtnorm)
  theta<- matrix(NA, nrow = nIter+1, ncol = dim(sigma)[1])
  theta[1,]<- theta_0
  rej_rate<-0
  for(i in 2:(nIter+1)){
    new_theta<- rmvnorm(1, mean = theta[i-1,], sigma = constant*sigma)
    cur_theta<-t(as.matrix(theta[i-1,]))
    U<-runif(1,0,1)
    num<-logPostFunc(new_theta,...)+dmvnorm(cur_theta, mean =new_theta, sigma = sigma, log = TRUE)
    den<-logPostFunc(cur_theta,...)+dmvnorm(new_theta, mean =cur_theta, sigma = sigma, log = TRUE)

    if(U<min(1,exp(num-den))){
      theta[i,] <-new_theta
    }else{
      theta[i,] <-cur_theta
      rej_rate  <-rej_rate+1
    }}


  myres<- list(theta= theta, rej_rate= rej_rate/nIter)
  return(myres)
}


res<-metropl(logPostFunc=LogPostPoisson,
                    theta_0= rep(0,nPara),
                    constant= 0.6,
                    sigma = J ,
                    priormu= mu_0,
                    priorsigma= Sigma_0,
                    X= X, Y=y,
                    nIter= 10000)


res[[2]]

compbetas<- data.frame(OptimCoefficients= as.vector(BetaCoef), MCMCCoef =colMeans(res[[1]]))
rownames(compbetas)<- covNames
compbetas
##d
Xpred <- matrix(c(1, 1, 1, 1, 0, 0, 0, 1, 0.5), nrow = 1)
predsamples <- rpois(10000, lambda = exp(Xpred %*% t(res[[1]])))
hist(predsamples, freq = FALSE)

##probability of 0
length(which(predsamples==0))/length(predsamples)
```