# Week 7 Lab Solutions – MAST90125: Bayesian Statistical learning

## Question One

Consider a Poisson regression,

$$y_i \sim \text{Pois}(\lambda_i) \quad \text{and} \quad \log(\lambda_i) = \mathbf{x}_i'\boldsymbol{\beta}, \quad \boldsymbol{\beta} \in \mathbb{R}^p$$

In lectures we learned various techniques for approximating the posterior distribution. In this lab, attempt as many of these techniques as possible to complete the following tasks.

Consider the dataset `Warpbreaks.csv`, which can be downloaded from Canvas. This dataset contains information of the number of breaks in a consignment of wool. In addition, Wool type (A or B) and tension level (L, M or H) are recorded. To investigate the association between the number of breaks and wool type, various forms of generalised linear model are proposed where Bayesian computing techniques should be used.

As a reminder the following techniques will be considered for approximating the posterior distribution.

- Metropolis-Hastings algorithm.
- Gibbs sampler.

When coding, assume the prior for the coefficients $\boldsymbol{\beta} \sim N(\mathbf{0}, 5\mathbf{I}_p)$.

Some hints:

An initial guess can be determined from fitting a Poisson regression using the function `glm`. Treat wool type as a factor using the function `glm`

```
set.seed(123456)
warpbreak= read.csv(file = './warpbreaks.csv',header=TRUE)
#This line will need to be changed when you run this yourself.
mod<-glm(breaks~as.factor(wool),data=warpbreak,family='poisson')
summary(mod)
```

```
##
## Call:
## glm(formula = breaks ~ as.factor(wool), family = "poisson", data = warpbreak)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        3.43518    0.03454  99.443  < 2e-16 ***
## as.factor(wool)B -0.20599    0.05157  -3.994 6.49e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
```

```
##     Null deviance: 297.37  on 53  degrees of freedom
## Residual deviance: 281.33  on 52  degrees of freedom
## AIC: 560
##
## Number of Fisher Scoring iterations: 4
```

```
Sigma <-vcov(mod); Sigma
```

```
##                    (Intercept) as.factor(wool)B
## (Intercept)        0.001193293     -0.001193293
## as.factor(wool)B -0.001193293      0.002659566
```

```
X<-model.matrix(mod)
```

### Metropolis-Hastings code

```
#Part one: function for performing Metropolis-Hastings sampling for
#Poisson regression. Proposed transition distribution is normal,
#with mean = betahat and variance-covariance matrix c^2*Sigma.
#Namely,Q(theta(t)|theta(t-1)) = N(theta(t)|thetamean=betahat, c^2*Sigma)
#Inputs:
#y: vector of responses
#X: predictor matrix including intercept.
#c: scalar associated with the variance-covariance matrix,
#thetamean: mean vector for the proposed transition distribution Q.
#Sigma: variance covariance matrix parameter in Q
#iter: number of iterations
#burnin: number of initial iterations to throw out.
library(mvtnorm)
```

```
## Warning: package 'mvtnorm' was built under R version 4.3.1
```

```
MetropolisHastings.fn<-function(y,X,c,thetamean,Sigma,iter,burnin){
p <-dim(X)[2]     #number of parameters

theta0<-rnorm(p)  #initial values
thetas<-matrix(0,iter,p)  #matrix to store values.
thetas[1,]<-theta0
indi<-0
for(i in 1:(iter-1)){
theta_t <-rmvnorm(1,mean=thetamean,sigma=c^2*Sigma) #draw candidate
theta_t <-as.numeric(theta_t)
xbc       <-X%*%theta_t
p.c        <-exp(xbc)  #Calculate lambda for candidate.
xb         <-X%*%thetas[i,]
p.b        <-exp(xb)    #Calculate lambda for theta(t-1)
#Note for code to work correctly, sum goes over only the dpois part because dpois
#evaluated over multiple observations is a vector but the dmvnorm
#for candidate/previous iteration is a scalar.
r_up<-sum(dpois(y,lambda=p.c,log=TRUE))+sum(dnorm(theta_t,0,sqrt(5),log=TRUE)) +
  dmvnorm(thetas[i,],mean=thetamean,sigma=c^2*Sigma,log=TRUE)
 #log joint dist + log proposal at the previous state
r_low<-sum(dpois(y,lambda=p.b,log=TRUE))+sum(dnorm(thetas[i,],0,sqrt(5),log=TRUE)) +
  dmvnorm(theta_t,mean=thetamean,sigma=c^2*Sigma,log=TRUE)
  #log joint dist + log proposal for the candidate state.
r <-r_up-r_low  #difference of log acceptance rate
#Draw an indicator whether to accept/reject candidate
ind<-rbinom(1,1,exp( min(c(r,0)) ) )
thetas[i+1,]<- ind*theta_t + (1-ind)*thetas[i,]
indi<-indi+ind
}
indi
#discard initial iterations
results<-thetas[-c(1:burnin),]
names(results)<-c('beta0','beta1') #column names
return(results)
}
```

```r
#formatting data into the correct format.

#one way to determine betaest and Sigma
# betaest<-mod$coef
# Sigma=vcov(mod)
# mod$coef
# Sigma
#another way to determine betaest and Sigma
modest <-lm(log(breaks)~as.factor(wool),data=warpbreak)
betaest<-modest$coef
Sigma <-vcov(modest)
betaest
```

```
##      (Intercept) as.factor(wool)B
##        3.3174392       -0.1521536
```

```r
Sigma
```

```
##                   (Intercept) as.factor(wool)B
## (Intercept)       0.006982306     -0.006982306
## as.factor(wool)B -0.006982306      0.013964613
```

```r
attempt2<-MetropolisHastings.fn(y=warpbreak$breaks,X=X,c=1,thetamean=betaest,Sigma=Sigma,
                                iter=10000,burnin=2000)

#Posterior means
colMeans(attempt2)
```

```
## [1]  3.4298672 -0.2001627
```

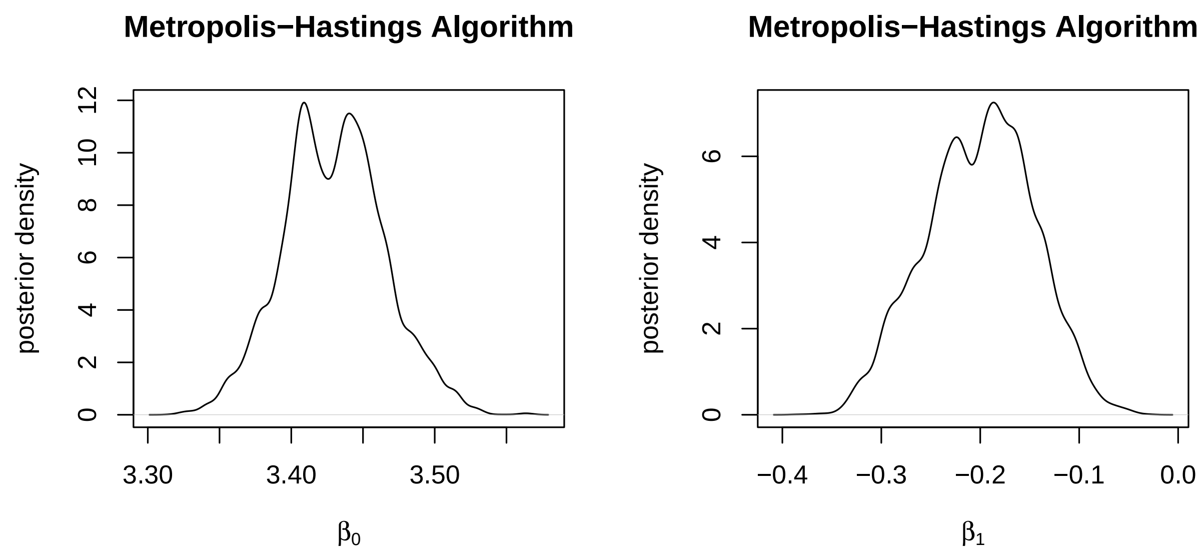
```r
#Posterior standard deviations
apply(attempt2,2,FUN=sd)
```

```
## [1] 0.03486872 0.05430271
```

```r
#95 % central credible intervals
apply(attempt2,2,FUN=function(x) quantile(x,c(0.025,0.975)) )
```

```
##            [,1]       [,2]
## 2.5%  3.363289 -0.3020488
## 97.5% 3.501333 -0.1008965
```

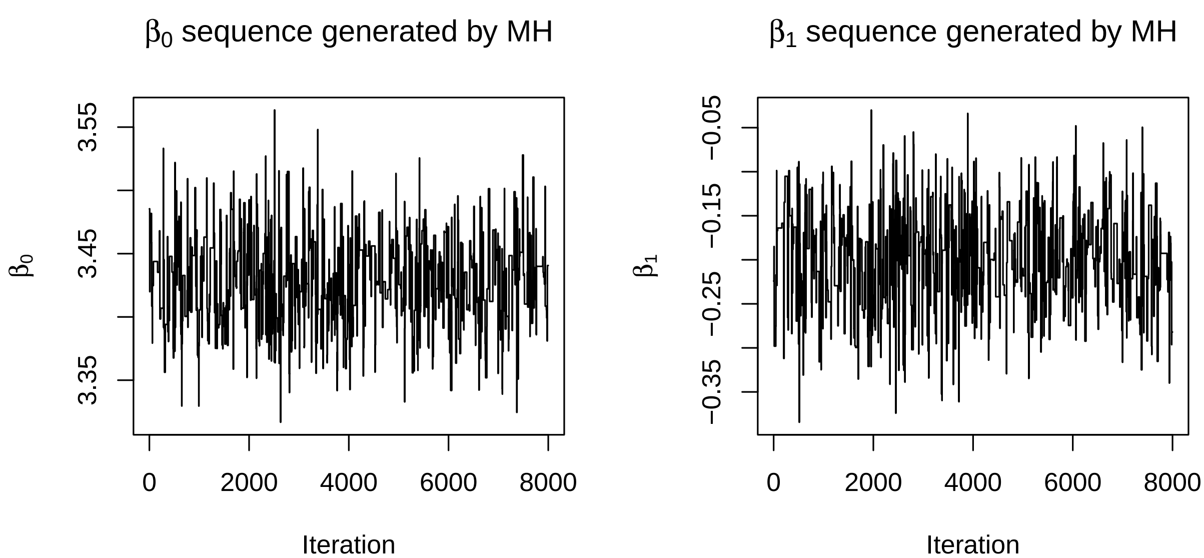
```r
par(mfrow=c(1,2))
#Plot marginal posteriors
plot(density(attempt2[,1]),type='l',xlab=expression(beta[0]),ylab='posterior density',
     main='Metropolis-Hastings Algorithm')
plot(density(attempt2[,2]),type='l',xlab=expression(beta[1]),ylab='posterior density',
     main='Metropolis-Hastings Algorithm')
```



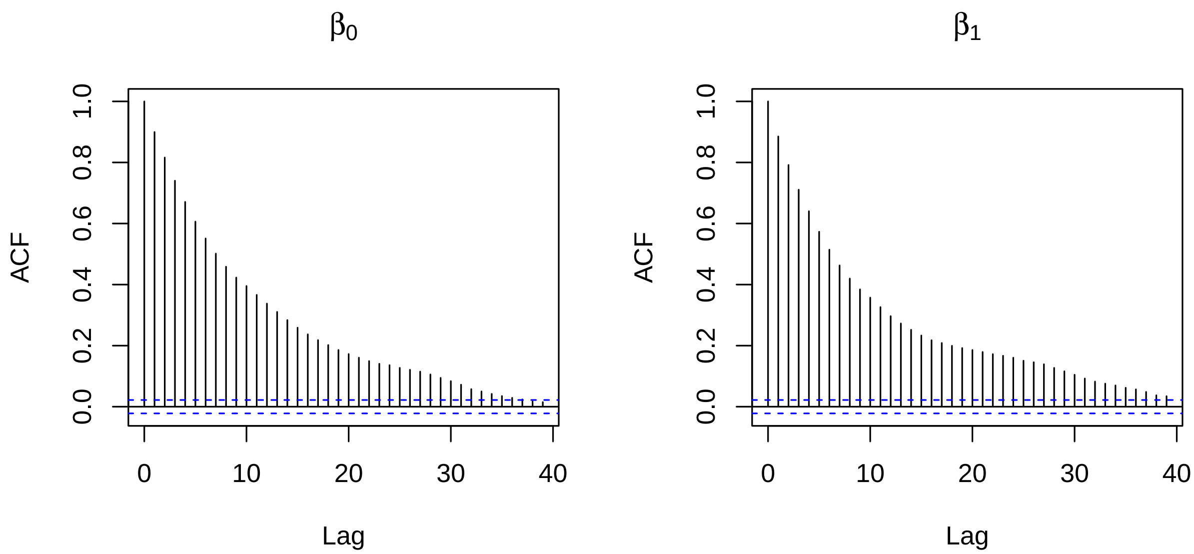

```
length(unique(attempt2[,1]))
```

```
## [1] 699
```

```
#Plot the simulated beta samples
plot(attempt2[,1], xlab='Iteration', ylab=expression(beta[0]), type='l',
     main=expression(paste(beta[0], ' sequence generated by MH')))
plot(attempt2[,2], xlab='Iteration', ylab=expression(beta[1]), type='l',
     main=expression(paste(beta[1], ' sequence generated by MH')))
```



```r
#ACF plot
acf(attempt2[,1], main=expression(beta[0]), cex.main=2)
acf(attempt2[,2], main=expression(beta[1]), cex.main=2)
```




**Gibbs sampler**

It can be shown that the posterior pdf of $\boldsymbol{\beta} = (\beta_0, \beta_1)'$ is

$$p(\beta_0, \beta_1 | (y_1, x_1), \cdots, (y_n, x_n)) \propto \exp\{(\sum_i y_i)\beta_0 + (\sum_i y_i x_i)\beta_1 - 0.1(\beta_0^2 + \beta_1^2) - e_0^{\beta}(\sum_i e^{x_i \beta_1})\}.$$

Thus, it is difficult to find the conditional posterior pdf of $\beta_0$ given $\beta_1$ and that of $\beta_1$ given $\beta_0$. Therefore, Gibbs sampler is not a good method for the problem considered here.