Assignment 3 Question 1 MAST90125: Bayesian Statistical Learning

Due: Friday 25 October 2019

There are places in this assignment where R code will be required. Therefore set the random seed so assignment is reproducible.

```
set.seed(695281)
library(mvtnorm) #Please change random seed to your student id number.
```

```
# Read  data
WOOL <- read.csv("Warpbreaks.csv")
```

```
# model poisson regression
mod<-glm(breaks~ ., WOOL, family = poisson(link = "log"))
X <- model.matrix(mod)
# sigma  <-vcov(mod)
y <- WOOL$breaks
p <-dim(X)[2]   #number of parameters
M <- 5*crossprod(X)
```

```
#Part one: function for performing Hamiltonian Monte Carlo for logistic regression.
#Inputs:
#y: vector of responses
#n: vector (or scalar) of trial sizes.
#X: predictor matrix including intercept.
#L: number of leapfrog steps.
#M is variance covariance matrix for normal prior of momentum variable \phi. Ideally diagonal.
#iter: number of iterations
#burnin: number of initial iterations to throw out.
HMC.fn<-function(y,X,L,M,iter,burnin){
p <-dim(X)[2]   #number of parameters
library(mvtnorm)
theta0<-rnorm(p) #initial values of beta
theta.sim<-matrix(0,iter,p+1) #matrix to store iterations plus acceptance.
theta.sim[1,1:p]<-theta0       #initial values in matrix.
epsilon<-1/L                 #epsilon assuming epsilon*L =1.
Minv   <-solve(M)


for(i in 1:(iter-1)){
phi       <-rmvnorm(1,mean=rep(0,p),sigma=M)   #draw momentum variable.
phi       <-as.numeric(phi)
phi0      <-phi                              #saving starting phi for calculation of r.
theta     <-theta.sim[i,1:p]                  #current state of theta.

lbd.b         <-exp(X%*%theta)  #calculate lambda.
gradtheta <- crossprod(X,y-lbd.b )   #Gradient of posterior = joint distribution with respect to theta.

#leapfrog steps.
```

```
for(j in 1:L){
  phi   <- phi + 0.5*epsilon*gradtheta   #first half step for phi
  theta <- theta + epsilon*(Minv%*%phi)  #full step for theta

lbd.c         <-exp(X%*%theta) #calculate probabilities of success at candidate (sub) state.
gradtheta <- crossprod(X,y-lbd.c )   #Gradient of posterior = joint distribution with respect to theta.

phi   <- phi + 0.5*epsilon*gradtheta #second half step for phi.
phi   <- as.numeric(phi)
}

#difference of log joint distributions at final iteration of leap.frog vs current state.
r<-sum( dpois(y,lambda = lbd.c,log=TRUE))+dmvnorm(phi,mean=rep(0,p),sigma=M,log=TRUE)-sum(dpois(y,lambda
#Draw an indicator whether to accept/reject candidate
ind<-rbinom(1,1,exp( min(c(r,0)) ) )
theta.sim[i+1,1:p]<- ind*theta + (1-ind)*theta.sim[i,1:p]
theta.sim[i+1,p+1] <- ind
}

#Removing the iterations in burnin phase
results<-theta.sim[-c(1:burnin),]
names(results)<-c('beta0','beta1','beta2','beta3','accept') #column names

return(results)
}
```

```
# L = 2
HMCl2<-HMC.fn(y=y,X=X,L=2,M=M,iter=10000,burnin=3000)

# L = 3
HMCl3<-HMC.fn(y=y,X=X,L=3,M=M,iter=10000,burnin=3000)

# L = 4
HMCl4<-HMC.fn(y=y,X=X,L=4,M=M,iter=10000,burnin=3000)
```

For L = 2

```
#Posterior means of beta0, beta1, beta2, beta3 Acceptance rate comparison

colMeans(HMCl2)
```

```
## [1]  3.1708593 -0.2058539  0.5199441  0.1978643  0.8324286
```

```
#Posterior standard deviations
apply(HMCl2,2,FUN=sd)
```

```
## [1] 0.05514903 0.05127605 0.06304162 0.06656290 0.37351195
```

```
#90 % credible intervals
apply(HMCl2,2,FUN=function(x) quantile(x,c(0.05,0.95)) )
```

```
##         [,1]       [,2]      [,3]      [,4] [,5]
## 5%  3.080528 -0.2897791 0.4157289 0.0861268    0
## 95% 3.261829 -0.1200403 0.6210818 0.3042288    1
```

```
# acceptance rate
colMeans(HMCl2)[5]
```

```
## [1] 0.8324286
```

For L = 3

```
#Posterior means of beta0, beta1, beta2, beta3 Acceptance rate comparison

colMeans(HMC13)
```

```
## [1]  3.1716852 -0.2064958  0.5193036  0.1979674  0.9230000
```

```
#Posterior standard deviations
apply(HMC13,2,FUN=sd)
```

```
## [1] 0.05701033 0.05164100 0.06314559 0.06958419 0.26661049
```

```
#90 % credible intervals
apply(HMC13,2,FUN=function(x) quantile(x,c(0.05,0.95)) )
```

```
##         [,1]       [,2]      [,3]       [,4] [,5]
## 5%  3.078909 -0.2911735 0.4148381 0.08240899    0
## 95% 3.266748 -0.1206554 0.6212428 0.31180073    1
```

```
# acceptance rate
colMeans(HMC13)[5]
```

```
## [1] 0.923
```

For L = 4

```
#Posterior means of beta0, beta1, beta2, beta3 Acceptance rate comparison
colMeans(HMC14)
```

```
## [1]  3.1719214 -0.2058379  0.5189682  0.1968202  0.9552857
```

```
#Posterior standard deviations
apply(HMC14,2,FUN=sd)
```

```
## [1] 0.05552354 0.05217846 0.06379348 0.06736974 0.20669064
```

```
#90 % credible intervals
apply(HMC14,2,FUN=function(x) quantile(x,c(0.05,0.95)) )
```

```
##         [,1]       [,2]      [,3]       [,4] [,5]
## 5%  3.080650 -0.2937332 0.4143111 0.08504399    1
## 95% 3.261597 -0.1192360 0.6239704 0.30748666    1
```

```
# acceptance rate
colMeans(HMC14)[5]
```

```
## [1] 0.9552857
```

The 90% credible interval tells there is 90% probability that parameter we seek to estimate is between the lower and upper bound of the given interval.

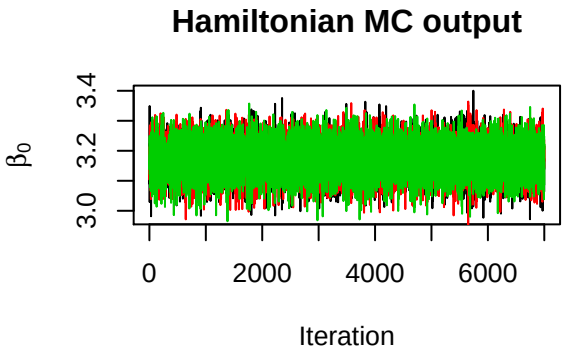
---

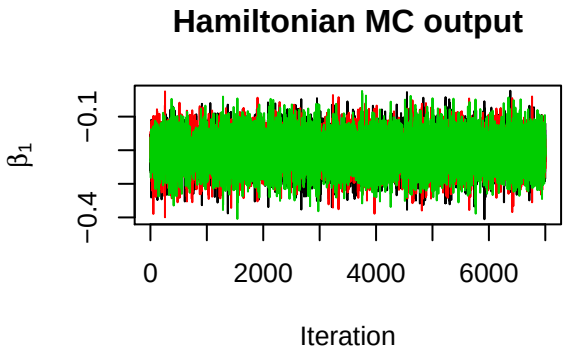
part D

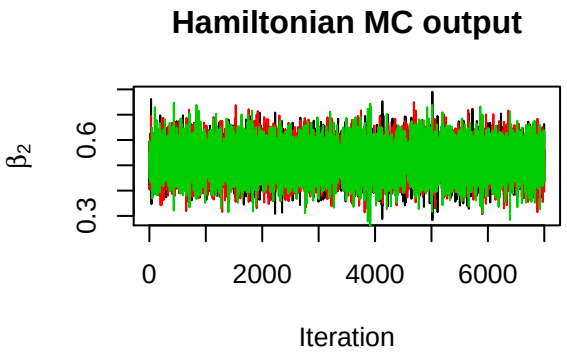
Answer: We will visually check if the chains converged to the same distribution

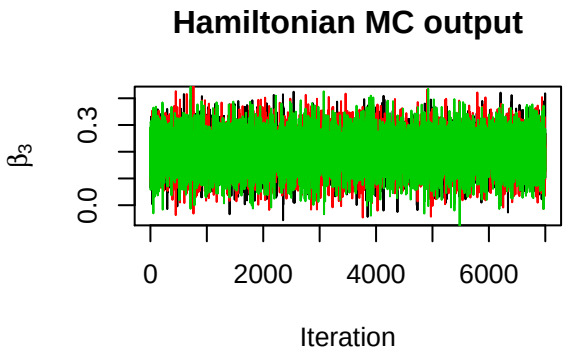
```
#plotting HMC.
par(mfrow=c(2,2))
for (i in 1:4) {
plot(HMC12[,i],type='l',main='Hamiltonian MC output',xlab='Iteration',ylab=bquote( beta[.(i-1)] ))
```

```
lines(HMCl3[,i],col=2)
lines(HMCl4[,i],col=3)
}
```









Checking the Gelman-Rubin diagnostic.

```
library(coda)
hl1<-as.mcmc.list(as.mcmc((HMCl2[1:3500,1:4])))
hl2<-as.mcmc.list(as.mcmc((HMCl2[1:3500,1:4])))
hl3<-as.mcmc.list(as.mcmc((HMCl3[1:3500,1:4])))
hl4<-as.mcmc.list(as.mcmc((HMCl3[3500+1:3500,1:4])))
hl5<-as.mcmc.list(as.mcmc((HMCl4[3500+1:3500,1:4])))
hl6<-as.mcmc.list(as.mcmc((HMCl4[3500+1:3500,1:4])))
hl<-c(hl1,hl2,hl3,hl4,hl5,hl6)

#Gelman-Rubin diagnostic.
gelman.diag(hl)[[1]]
```

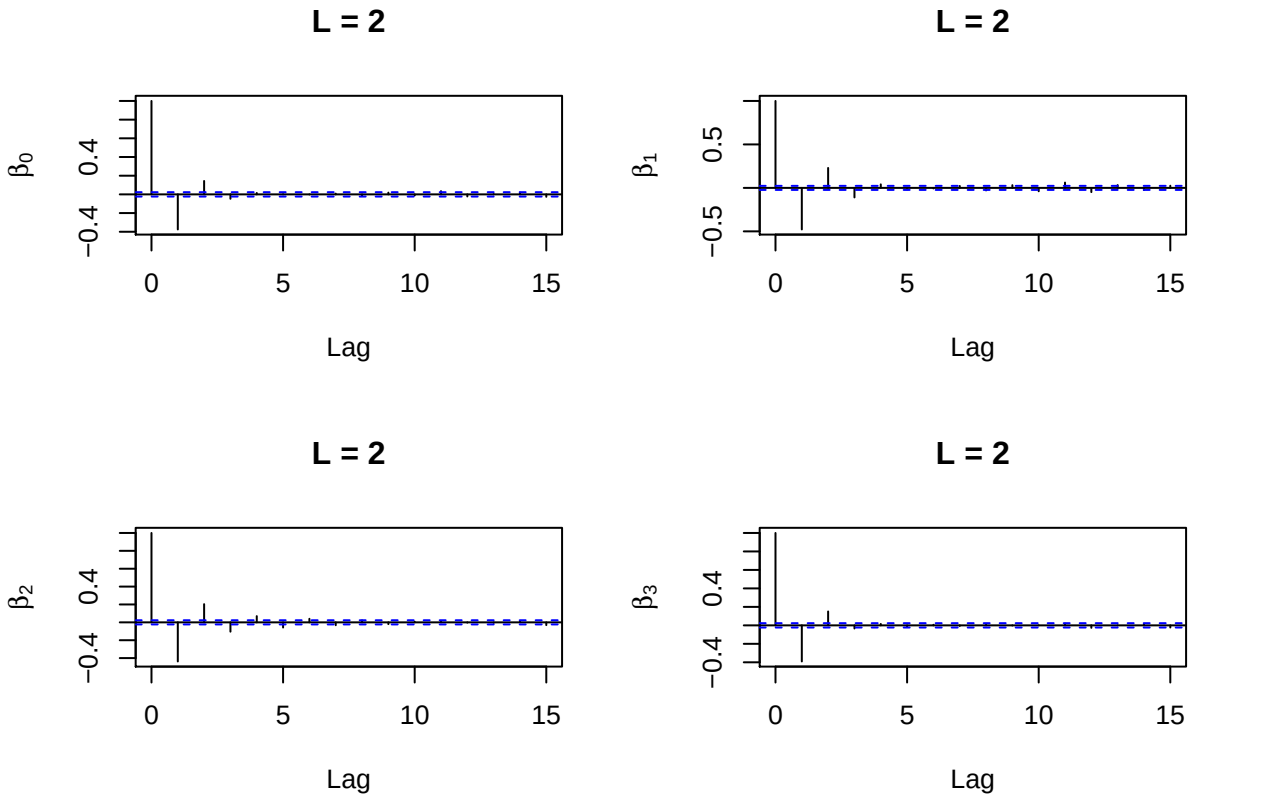
```
##      Point est. Upper C.I.
## [1,]  0.9998943   1.000006
## [2,]  1.0001577   1.000182
## [3,]  1.0002284   1.000370
## [4,]  1.0001417   1.000292
```

```
#effective sample size.
effectiveSize(hl)
```

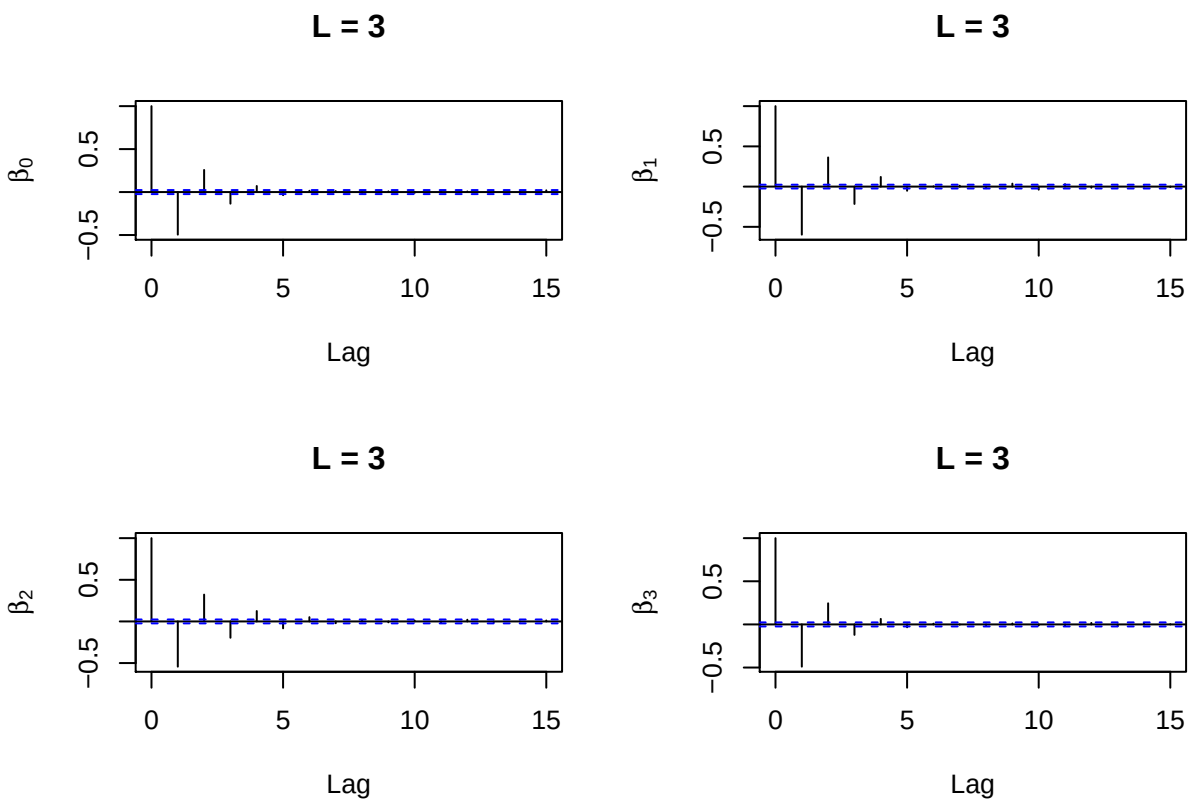
```
##     var1     var2     var3     var4
## 55123.72 87829.08 67323.85 56010.16
```

Check the acf plots

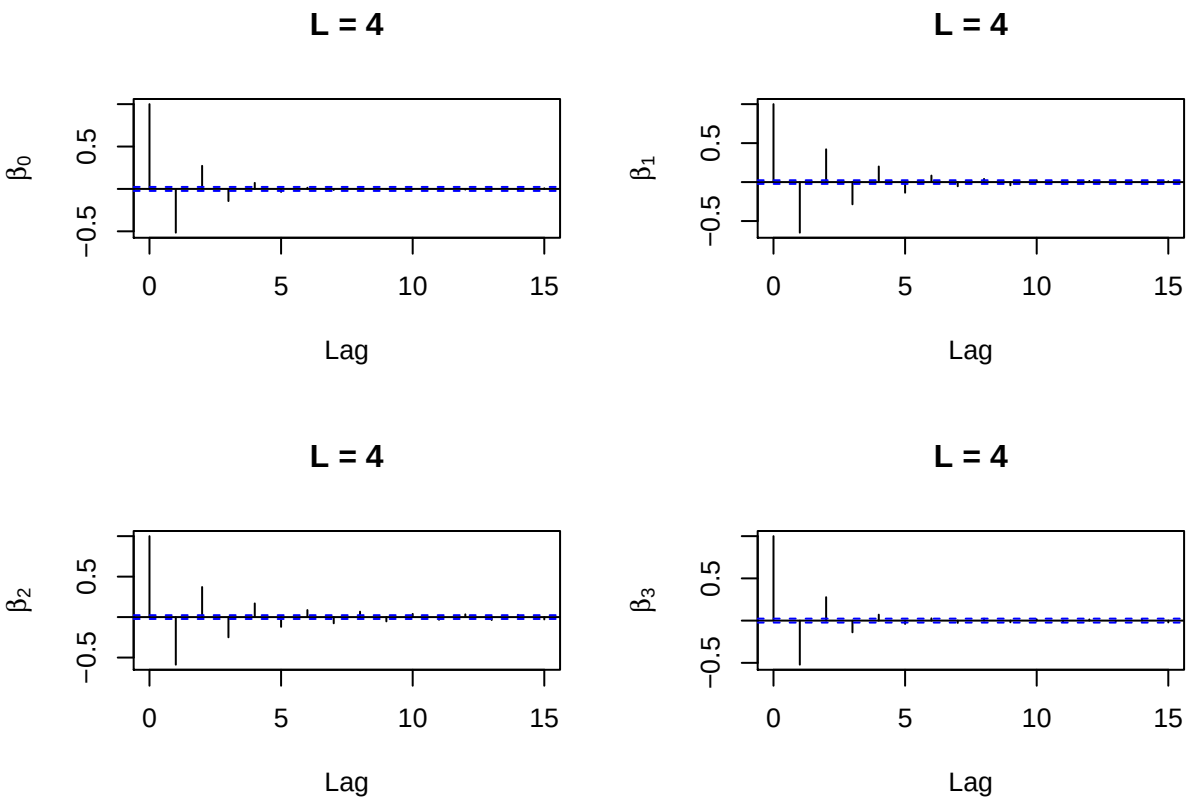
```r
# L=2
par(mfrow=c(2,2))
for (i in 1:4) {
acf(HMCl2[,i],ylab=bquote( beta[.(i-1)] ),lag.max = 15,main="L = 2")
}
```




```r
# L=3
par(mfrow=c(2,2))
for (i in 1:4) {
acf(HMCl3[,i],ylab=bquote( beta[.(i-1)] ),lag.max = 15,main="L = 3")
}
```




```
# L=4
par(mfrow=c(2,2))
for (i in 1:4) {
acf(HMC14[,i],ylab=bquote( beta[.(i-1)] ),lag.max = 15,main="L = 4")
}
```


L = 4
β0
0.5
−0.5
0
5
10
15
Lag
L = 4
β1
0.5
−0.5
0
5
10
15
Lag
L = 4
β2
0.5
−0.5
0
5
10
15
Lag
L = 4
β3
0.5
−0.5
0
5
10
15
Lag