# Steven Maharaj 695281 Assignment 2, Question 3 MAST90125: Bayesian Statistical Learning

**Due: Friday 20 September 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)  #Please change random seed to your student id number.
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(tidyr)
library(TruncatedNormal)
library(mvtnorm)
```

```
##
## Attaching package: 'mvtnorm'

## The following objects are masked from 'package:TruncatedNormal':
##
##     pmvnorm, pmvt
```

```
library(coda)
```

```
rtn <- function(n,b,a,mu,Sigma){
  u <- runif(n)
  g <- pnorm((b-mu)/Sigma) - pnorm((a-mu)/Sigma)
  x <-qnorm((g) * u + pnorm((a-mu)/Sigma))*Sigma + mu
}
```

---

PART C

We mplement a Gibbs sampler to fit the same mixed model, but now with a probit link.

Assumsing,

- $p(\boldsymbol{\beta}) \propto 1$
- $p(\boldsymbol{u}) = \mathcal{N}(\mathbf{0}, \sigma_u^2 \boldsymbol{I})$
- $p(\tau_u) = Ga(\alpha_u, \gamma_u)$

It can be shown that we have the folling conditional posteriors

$$p\left(\tau_u|\cdot\right) = \mathrm{Ga}\left(\alpha_u + q/2, \gamma_u + \mathbf{u}'\mathbf{u}/2\right)$$

$$p\left(\left(\begin{array}{c}\beta\\ u\end{array}\right)|\cdot\right) = \mathcal{N}\left(\left(\begin{array}{cc}X'X & X'Z\\ Z'X & Z'Z+\tau_u\boldsymbol{I}^{-1}\end{array}\right)^{-1}\left(\begin{array}{c}X'z\\ Z'z\end{array}\right), \left(\begin{array}{cc}X'X & X'Z\\ Z'X & Z'Z+\tau_u\boldsymbol{I}^{-1}\end{array}\right)^{-1}\right)$$

We define our inputs for the Gibbs Sampler

```
#Step one: Importing data,  constructing design matrices and calculating matrix dimensions.
dataX= read.csv("Contraceptionsubset.csv",header=TRUE)
n<-dim(dataX)[1]
Z    = table(1:n,dataX$district)     #incidence matrix for district
Q    = dim(Z)[2]
D1   = table(1:n,dataX$livch) #Dummy indicator for living children
D2   = table(1:n,dataX$urban) #Dummy indicator for urban status

#fixed effect design matrix
X    = cbind(rep(1,n),dataX$age,D1[,-1],D2[,-1])
P    = dim(X)[2]
y    = rep(0,n)
y[dataX$use %in% 'Y'] = 1
a <- 0.001
g <- 0.001
# iter =2000
```

Construct a Gibbs sampler

```
Gibbsq3 <- function(iter,Z,X,y,burnin,tauu_0,a,g){
  q    = dim(Z)[2]
  p    = dim(X)[2]

  W<-cbind(X,Z)            #for the joint conditional posterior for b,u
  WTW <-crossprod(W)
  I0 <- diag(p+q)
  diag(I0)[1:p] <- 0

  #starting values.
  t_u <- tauu_0
  u   <-rnorm(q,0,sd=1/sqrt(t_u))

  uTu <- crossprod(u)

  #storing results.
  par <-matrix(0,iter,p+q+1)

  for (i in 1:iter) {
    Prec <-WTW + t_u*I0

    t_u <- rgamma(1,a + q*0.5,g + uTu*0.5)
    z <- rtn(n=length(y),b=100,a=0,mu = 0.5,Sigma = 1)*(y==1) + rtn(n=length(y),b=0,a=-100,mu = 0.5,Sig
    P.mean <- solve(Prec)%*%crossprod(W,z)
    P.var  <-solve(Prec)
    res <- rmvnorm(1,P.mean,P.var)
    b <- res[1:p]
```

```
    u <- res[p+1:q]
    par[i,]<-c(b,u,1/t_u)
  }
par <-par[-c(1:burnin),] #removing initial iterations
colnames(par)<-c(paste('beta',1:p,sep=''),paste('u',1:q,sep=''),"sigma2_u")
 return(par)
}
```

```
chain1 <- Gibbsq3(iter=2000,Z=Z,X=X,y=y,burnin=1000,tauu_0 = 1,a=a,g=g)
chain2 <- Gibbsq3(iter=2000,Z=Z,X=X,y=y,burnin=1000,tauu_0 = 0.5,a=a,g=g)
chain3 <- Gibbsq3(iter=2000,Z=Z,X=X,y=y,burnin=1000,tauu_0 = 2,a=a,g=g)
chain4 <- Gibbsq3(iter=2000,Z=Z,X=X,y=y,burnin=1000,tauu_0 = 5,a=a,g=g)

half_len <-

ml1<-as.mcmc.list(as.mcmc((chain1[1:500,])))
ml2<-as.mcmc.list(as.mcmc((chain2[1:500,])))
ml3<-as.mcmc.list(as.mcmc((chain3[1:500,])))
ml4<-as.mcmc.list(as.mcmc((chain4[1:500,])))
ml5<-as.mcmc.list(as.mcmc((chain1[500+1:500,])))
ml6<-as.mcmc.list(as.mcmc((chain2[500+1:500,])))
ml7<-as.mcmc.list(as.mcmc((chain4[500+1:500,])))
ml8<-as.mcmc.list(as.mcmc((chain4[500+1:500,])))
estml<-c(ml1,ml2,ml3,ml4,ml5,ml6,ml7,ml8)
```

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

```
##          Point est. Upper C.I.
## beta1     1.1176444   1.120682
## beta2     0.9995164   1.000986
## beta3     1.0005415   1.003507
## beta4     1.0000740   1.002366
## beta5     1.0005339   1.003547
## beta6     0.9998045   1.001072
## u1        1.1235185   1.127058
## u2        1.1213258   1.124127
## u3        1.1226484   1.125437
## u4        1.1221924   1.125214
## u5        1.1222496   1.125479
## sigma2_u  1.4626253   3.065819
```

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

```
##      beta1      beta2      beta3      beta4      beta5      beta6
## 3951.26349 4000.00000 3784.14787 4487.51968 3773.86389 4000.00000
##         u1         u2         u3         u4         u5   sigma2_u
## 3945.24870 3546.29270 4032.19170 3953.00630 3650.83527   19.74059
```

```
#Reporting posterior means and credible intervals.
#Means
colMeans(rbind(chain1,chain2,chain3,chain4))
```

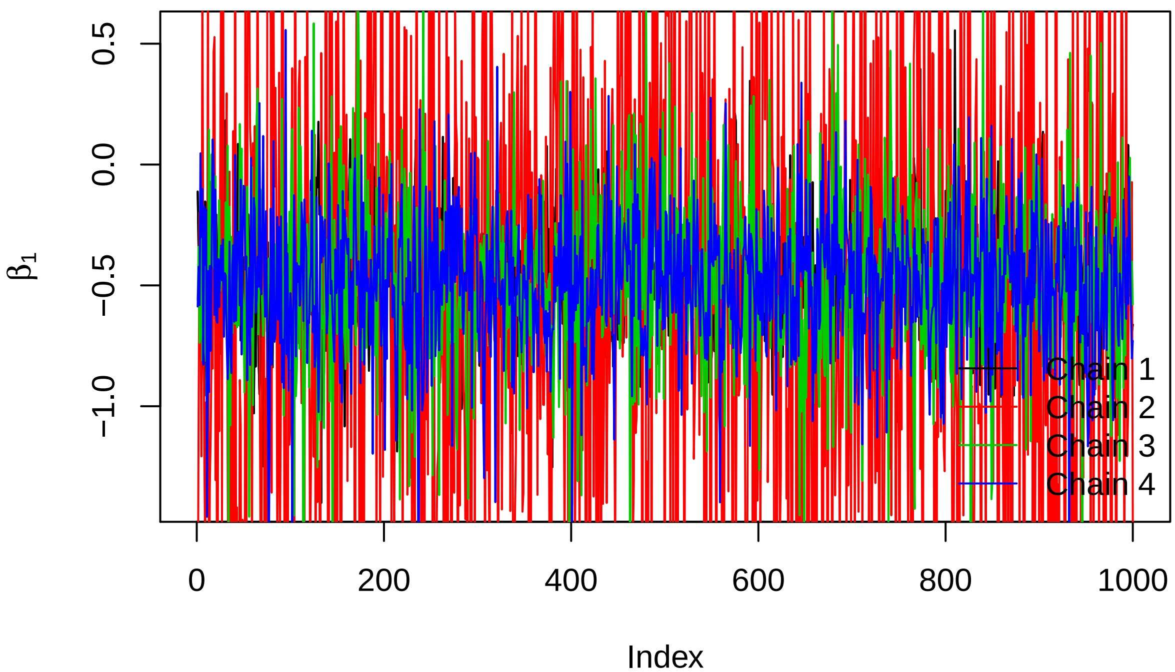
```
##       beta1       beta2       beta3       beta4       beta5       beta6
## -0.48173780 -0.01374893  0.39860266  0.47440551  0.58262533  0.40761171
```

```
##          u1          u2          u3          u4          u5    sigma2_u
## -0.35050056 -0.10285299  0.16369668  0.06085934  0.24688847  2.09196050
```

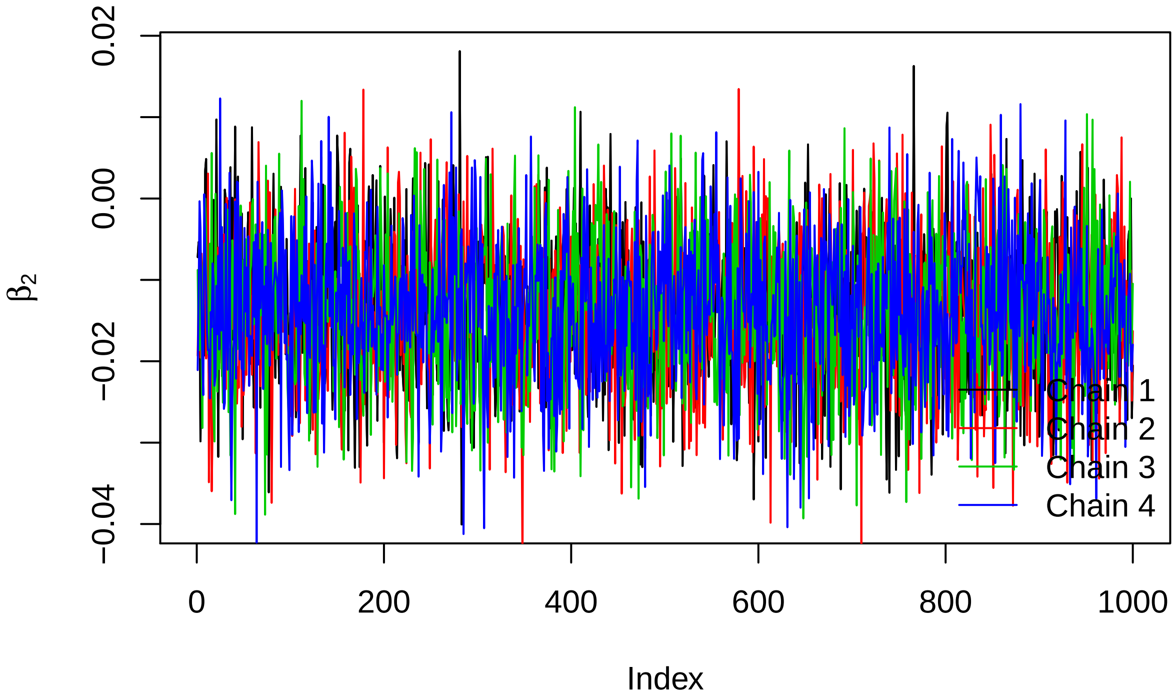
```r
#95 % central Credible interval
apply(rbind(chain1,chain2,chain3,chain4) ,2, FUN =function(x) quantile(x,c(0.025,0.975) ))
```

```
##          beta1        beta2      beta3     beta4     beta5     beta6
## 2.5%  -1.92424 -0.031562776 0.05176651 0.1110273 0.2165573 0.1391559
## 97.5%  0.92777  0.003840846 0.72746433 0.8295473 0.9406774 0.6716043
##              u1        u2        u3        u4        u5    sigma2_u
## 2.5%  -1.781493 -1.491865 -1.193218 -1.338907 -1.173781  0.06260205
## 97.5%  1.101170  1.303160  1.615135  1.539666  1.686422 13.27407538
```

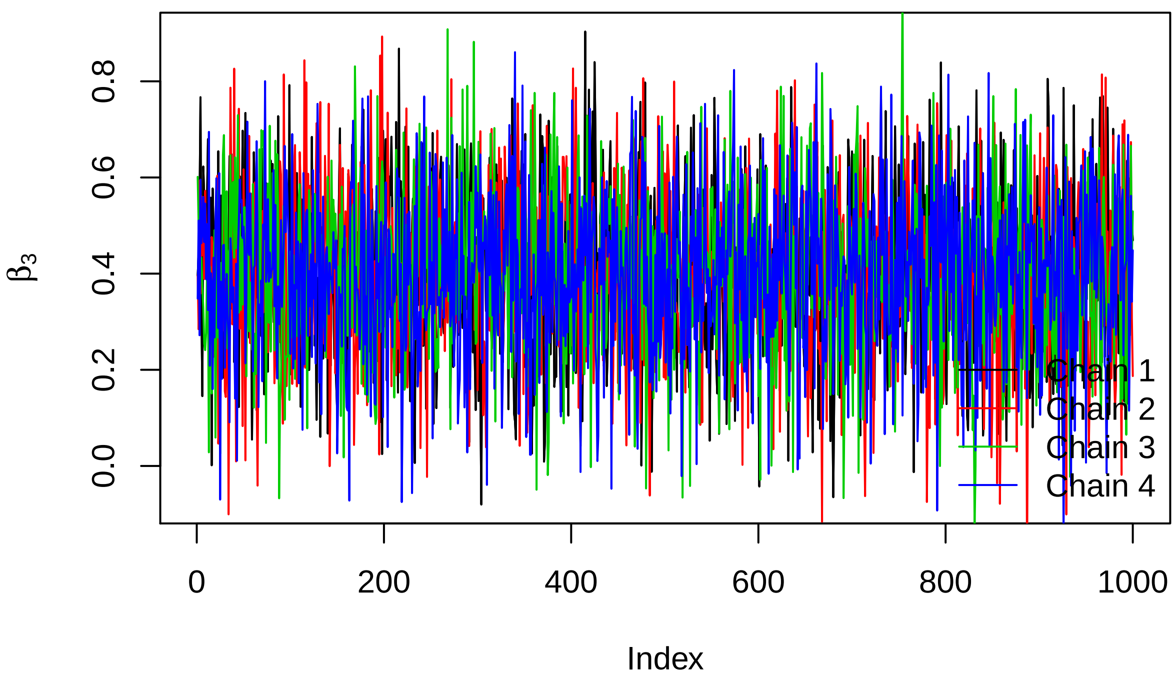
```r
# beta1
plot(chain1[,1],type='l',ylab=expression(beta[1]),col=1)
lines(chain1[,1],type='l',col=1,ylab=expression(beta[1]))
lines(chain2[,1],type='l',col=2,ylab=expression(beta[1]))
lines(chain3[,1],type='l',col=3,ylab=expression(beta[1]))
lines(chain4[,1],type='l',col=4,ylab=expression(beta[1]))
legend('bottomright',legend=c('Chain 1','Chain 2','Chain 3','Chain 4'),col=1:4,lty=1,bty='n')
```




```r
# beta2
plot(chain1[,2],type='l',ylab=expression(beta[2]),col=1)
lines(chain1[,2],type='l',col=1,ylab=expression(beta[2]))
lines(chain2[,2],type='l',col=2,ylab=expression(beta[2]))
lines(chain3[,2],type='l',col=3,ylab=expression(beta[2]))
lines(chain4[,2],type='l',col=4,ylab=expression(beta[2]))
legend('bottomright',legend=c('Chain 1','Chain 2','Chain 3','Chain 4'),col=1:4,lty=1,bty='n')
```

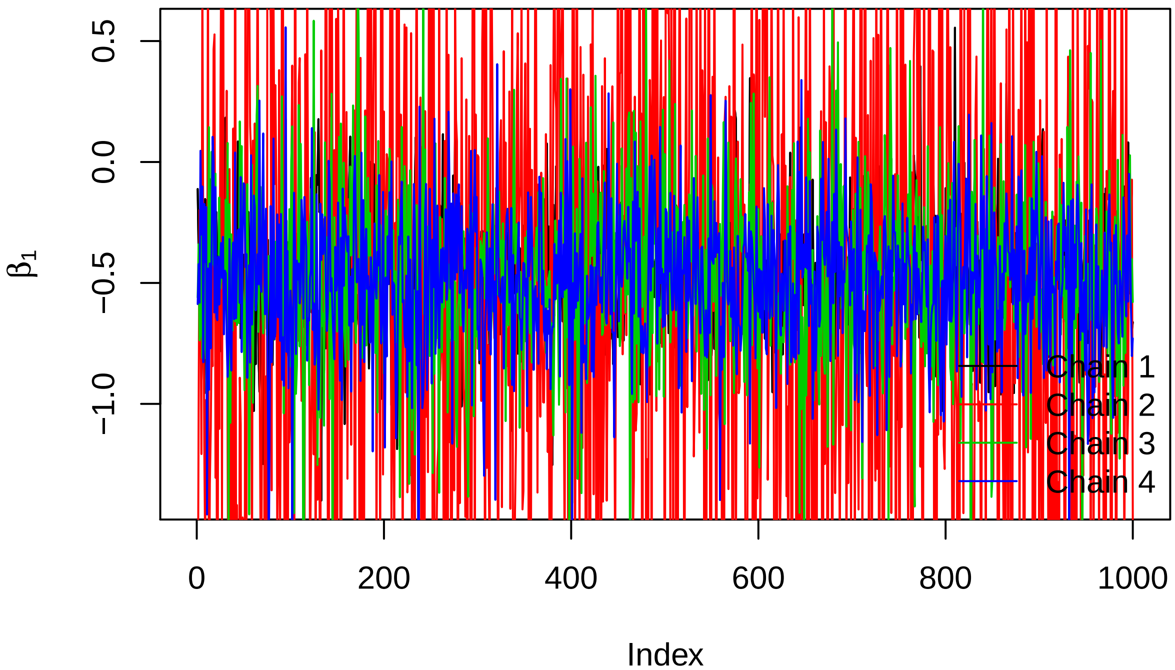


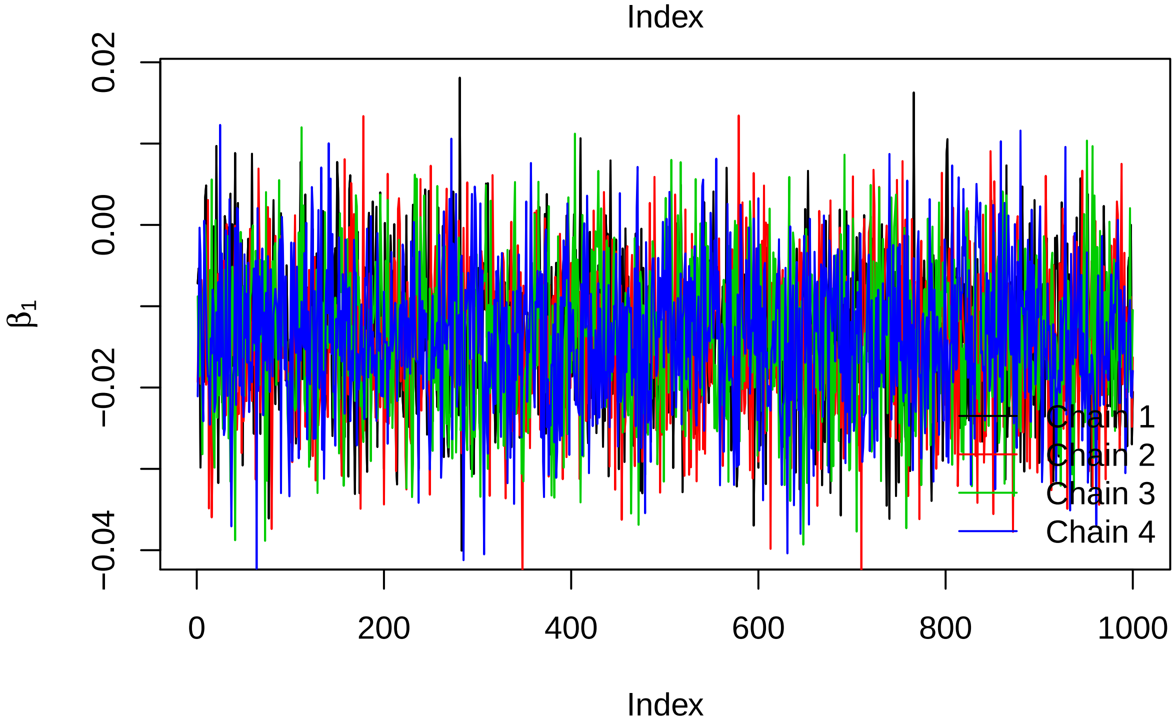

```r
# beta1
plot(chain1[,3],type='l',ylab=expression(beta[3]),col=1)
lines(chain1[,3],type='l',col=1,ylab=expression(beta[3]))
lines(chain2[,3],type='l',col=2,ylab=expression(beta[3]))
lines(chain3[,3],type='l',col=3,ylab=expression(beta[3]))
lines(chain4[,3],type='l',col=4,ylab=expression(beta[3]))
legend('bottomright',legend=c('Chain 1','Chain 2','Chain 3','Chain 4'),col=1:4,lty=1,bty='n')
```

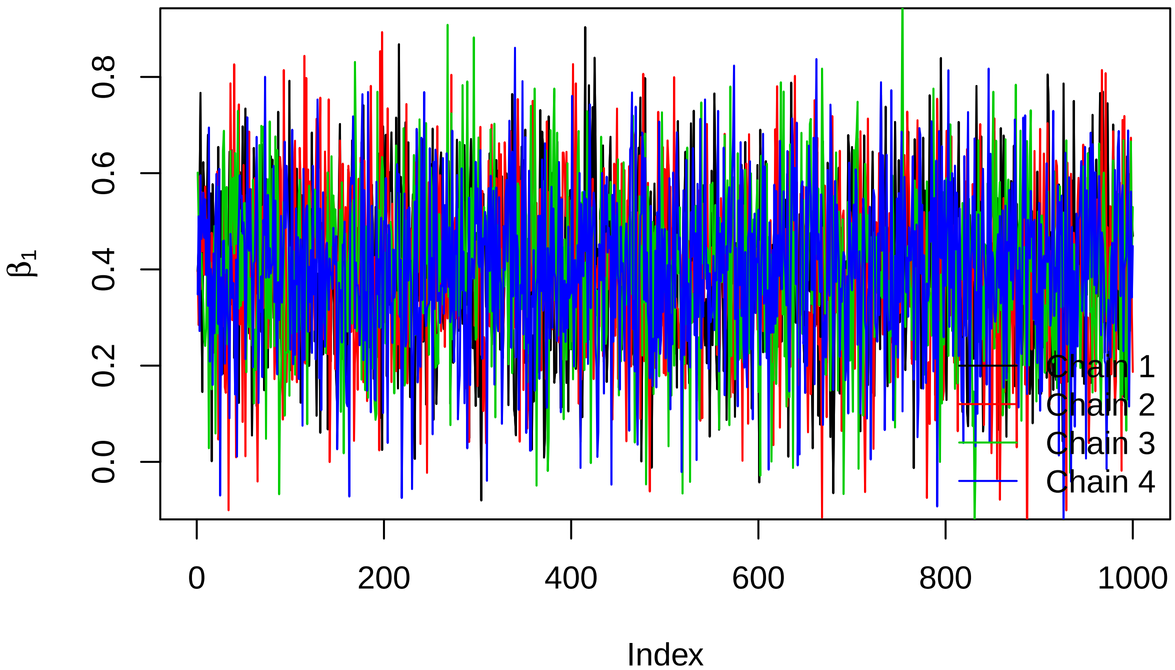


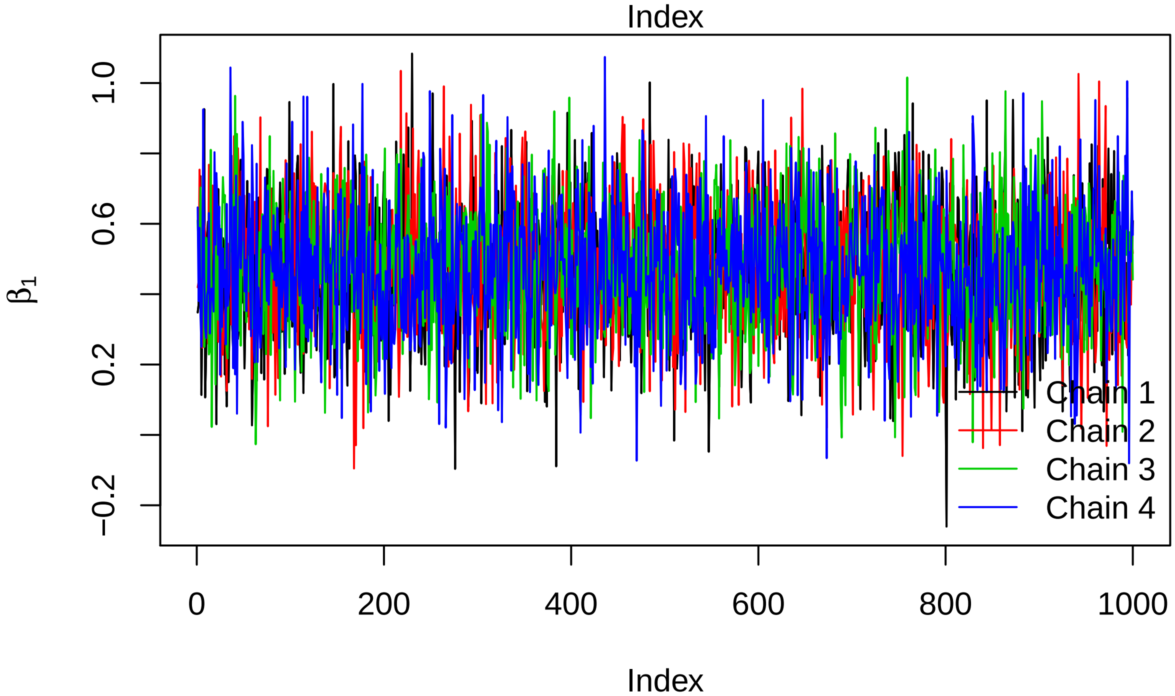

```r
for(i in 1:6){
  # beta1
  plot(chain1[,i],type='l',ylab=expression(beta[1]),col=1)
```

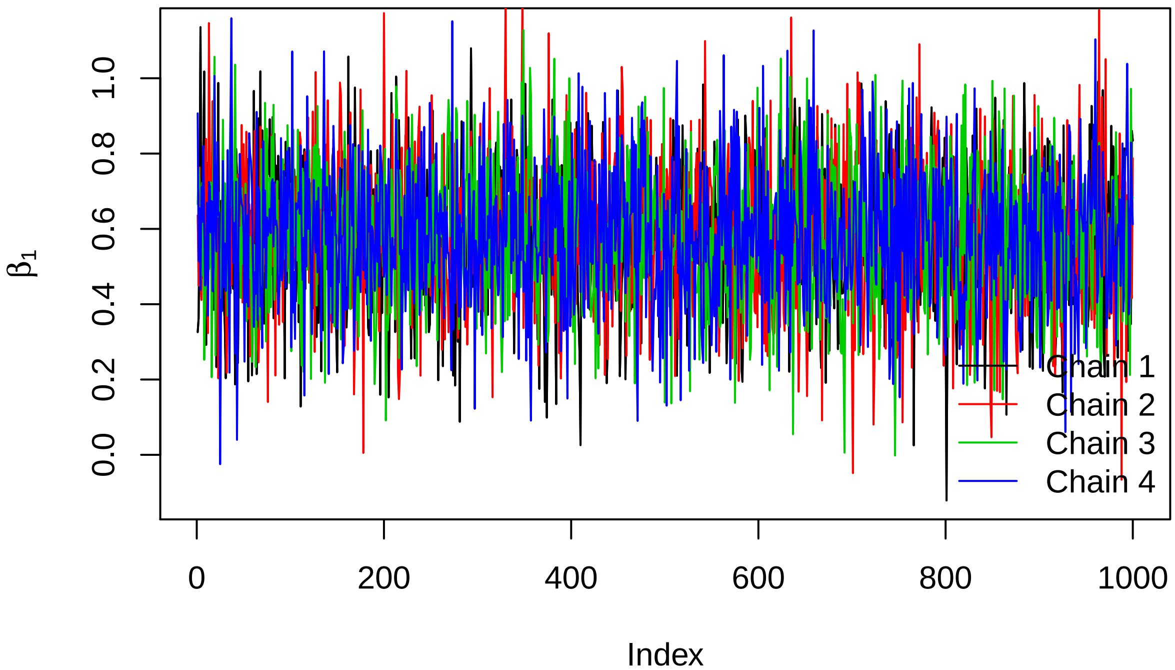
```
lines(chain1[,i],type='l',col=1,ylab=expression(beta[i]))
lines(chain2[,i],type='l',col=2,ylab=expression(beta[i]))
lines(chain3[,i],type='l',col=3,ylab=expression(beta[i]))
lines(chain4[,i],type='l',col=4,ylab=expression(beta[i]))
legend('bottomright',legend=c('Chain 1','Chain 2','Chain 3','Chain 4'),col=1:4,lty=1,bty='n')}
```



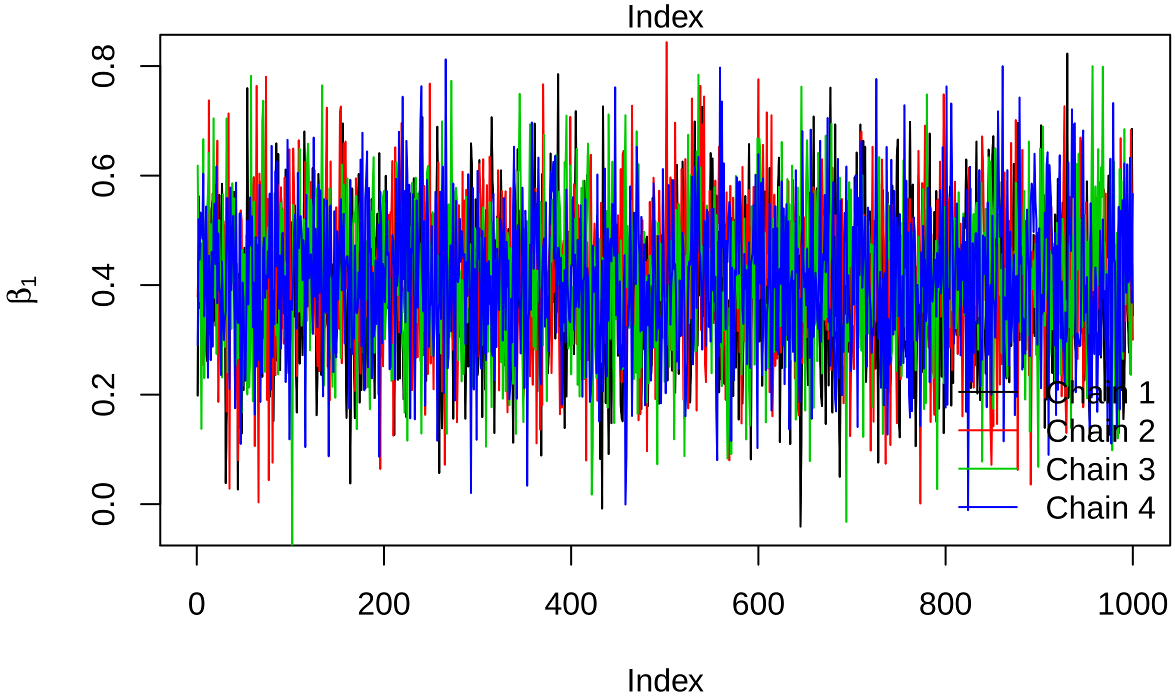




β1
0.0
0.2
0.4
0.6
0.8
0
200
400
600
800
1000
Index
Chain 1
Chain 2
Chain 3
Chain 4


β1
−0.2
0.2
0.6
1.0
0
200
400
600
800
1000
Index
Chain 1
Chain 2
Chain 3
Chain 4



1.0
0.8
0.6
0.4
0.2
0.0
$\beta_1$
0
200
400
600
800
1000
Index
Chain 1
Chain 2
Chain 3
Chain 4

0.8
0.6
0.4
0.2
0.0
$\beta_1$
0
200
400
600
800
1000
Index
Chain 1
Chain 2
Chain 3
Chain 4