Steven Maharaj 695281 Assignment 2, Question 2

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(mvtnorm)
library(coda)
library(ggplot2)
library(tidyr)
```

PART A

```
# Read the Data
Hiron <- read.csv("Hiron.csv")
HironHomo <- Hiron %>%filter(homc282y==1) %>% select(-idnum)
HironWild <- Hiron %>%filter(homc282y==0) %>% select(-idnum)
```

```
#fit linear regression using lm

model <- lm(logsf ~ time,data = HironHomo)
model$coefficients
```

```
## (Intercept)        time
##  4.23987253  0.07126085
```

```
intercept <-matrix(1,length(HironHomo$time),1)

X <- cbind(intercept,HironHomo$time)
XTX <- crossprod(X)
XTXinv <-solve(XTX)
# (XTX)^-1
XTXinv
```

```
##              [,1]          [,2]
## [1,]  0.033734956 -0.0015415009
## [2,] -0.001541501  0.0001062175
```

```r
#Estimated error variance
Sigma2<-sigma(model)^2
Sigma2
```

```
## [1] 1.715042
```

PART B

For This Question we define proper priors for $\boldsymbol{\beta}, \boldsymbol{\tau}$ using the results from part a. That is using the following formula.

$$p(\theta|\boldsymbol{y}) = \frac{p(\boldsymbol{y_2}|\theta)p(\theta|\boldsymbol{y_1})}{p(\boldsymbol{y_2}|\boldsymbol{y_1})}$$

Using the posterior from the previous part we let the prior for this part be

$$p(\boldsymbol{\beta}|\tau) = \mathcal{N}(\hat{\boldsymbol{\beta}}_1, (\boldsymbol{X}_1\boldsymbol{X}_1)^{-1}/\tau)$$
$$p(\boldsymbol{\beta}|\tau) = Ga(\frac{n_1 - p}{2}, \frac{(n_1 - p)s^2}{2})$$

where the subscript 1 indicates the results are from group 1 (analysed in 2a) alone. Using the results from lecture 13, we drop the prior for τ_β, and in all other places in the joint distribution, replace τ_β and τ_e with τ.

- $\boldsymbol{K} = (\boldsymbol{X}_1\boldsymbol{X}_1)^{-1}$
- $\boldsymbol{\beta}_0 = \hat{\boldsymbol{\beta}}_1$
- $\alpha = \frac{n_1 - p}{2}$
- $\gamma = \frac{(n_1 - p)s^2}{2}$

Thus we get the following conditional posteriors

$$p\left(\tau|\mathbf{y}, \boldsymbol{\beta}, \boldsymbol{\beta}_0, \mathbf{K}\right) = \text{Ga}\left(\alpha + \frac{n+p}{2}, \gamma + \frac{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + (\boldsymbol{\beta} - \boldsymbol{\beta}_0)'\,\mathbf{K}^{-1}\,(\boldsymbol{\beta} - \boldsymbol{\beta}_0)}{2}\right)$$

$$p\left(\boldsymbol{\beta}|\mathbf{y}, \boldsymbol{\beta}_0, \mathbf{K}, \tau\right) = \mathcal{N}\left(\left(\mathbf{X}'\mathbf{X} + \mathbf{K}^{-1}\right)^{-1}\left(\mathbf{X}'\mathbf{y} + \mathbf{K}^{-1}\beta_0\right), \left(\mathbf{X}'\mathbf{X} + \mathbf{K}^{-1}\right)^{-1}/\tau\right)$$

Below we fit a Bayesian regression using a Gibbs sampler to **only the wildtype (`homc282y=0`) data**.

```r
# define our Gibbs sampler function


#Arguments are
#X:  matrix of predictors dimension n times p with flat prior. Includes the intercept.
#Z:  matrix of predictors dimension n times q with normal prior for u.
#y:  response vector, length p.
#taue_0,tauu_0: initial value for the residual and random effect precision.
#a.u,b.u. Hyper-parameter for gamma prior for tau_u
#a.e,b.e. Hyper-parameter for gamma prior for tau_e
#iter: number of iterations
#burnin: number of initial iterations to remove
normalmm.Gibbs<-function(iter,Z,X,y,burnin,taue_0,tauu_0,a.u,b.u,a.e,b.e){
  n   <-length(y) #no. observations
  p   <-dim(X)[2] #no of fixed effect predictors.
  q   <-dim(Z)[2] #no of random effect levels
```

```
  tauu<-tauu_0
  taue<-taue_0
  #starting value for u.
  u0   <-rnorm(q,0,sd=1/sqrt(tauu))

  #Building combined predictor matrix.
  W<-cbind(X,Z)            #for the joint conditional posterior for b,u
  WTW <-crossprod(W)
  library(mvtnorm)

  #storing results.
  par <-matrix(0,iter,p+q+2)  #p beta coefficient, q u coefficients and 2 precision coefficient.

  #Create modified identity matrix for joint posterior.
  I0  <-diag(p+q)
  diag(I0)[1:p]<-0

  for(i in 1:iter){
    #Conditional posteriors.
    tauu <-rgamma(1,a.u+0.5*q,b.u+0.5*sum(u0^2)) #sample tau_u
    #Updating component of normal posterior for beta,u
    Prec <-WTW + tauu*I0/taue
    P.mean <- solve(Prec)%*%crossprod(W,y)
    P.var  <-solve(Prec)/taue
    betau <-rmvnorm(1,mean=P.mean,sigma=P.var) #sample beta, u
    betau <-as.numeric(betau)
    err   <- y-W%*%betau
    taue  <-rgamma(1,a.e+0.5*n,b.e+0.5*sum(err^2)) #sample tau_e
    #storing iterations for beta, u, and standard deviation of e, u.
    par[i,]<-c(betau,1/sqrt(tauu),1/sqrt(taue))
    u0     <-betau[p+1:q]  #extracting u so we can update tau_u.
  }

par <-par[-c(1:burnin),] #removing initial iterations
colnames(par)<-c(paste('beta',1:p,sep=''),paste('u',1:q,sep=''),'sigma_b','sigma_e')
 return(par)
}
```

```
# Define inputs for Gibbs sampler

Kinv <- XTX
X2 <- cbind(matrix(1,length(HironWild$time),1),HironWild$time)
y <- HironWild$logsf
b0 <- model$coefficients
n1 <-dim(X)[1]
p   <-dim(X2)[2]
a <-  (n1-p)*0.5
s2 <- sum((HironHomo$logsf - X%*%b0)^2)/(n1-p)
g <- (n1-p)*s2*0.5
```

```
Gibbsq2 <- function(iter,X,y,burnin,tau_0,a,g,b0,Kinv){
  n  <-length(y) #no. observations
  p   <-dim(X)[2] #no of fixed effect predictors.
  XXkii <- solve(crossprod(X) + Kinv)
```

```
  P.mean <- XXkii%*%(t(X)%*%y + Kinv%*%b0)

  b <- rnorm(p,0,sd=1/sqrt(tau_0))
  tau<- tau_0

  #storing results.
  par <-matrix(0,iter,p+3)

  for (i in 1:iter) {

    err <- y-X%*%b
    T_y <- sum(err^2)

    tau <-rgamma(1,a+(n+p)*0.5, g + 0.5*T_y + 0.5*t(b-b0)%*%Kinv%*%(b-b0) )
    b <- rmvnorm(1,mean=P.mean,sigma=XXkii/tau)
    b <-as.numeric(b)

    # posterior checking
  Xb <- X%*%as.vector(b)
  y_rep <- Xb+rnorm(length(y))*sqrt(1/tau)
  T_rep <- sum((y_rep-Xb)^2)

  #storing iterations for beta, tau,.
  par[i,] <- c(b[1],b[2],1/tau,T_y,T_rep)



  }

  par <-par[-c(1:burnin),] #removing initial iterations
  colnames(par)<-c("beta0","beta1","sigma2","T_y","T_rep")
  return(par)
}
```

```
chain1 <- Gibbsq2(iter=10000,X = X2,y=y,burnin=1000,tau_0=1,a=a,g=g,b0=b0,Kinv=Kinv)
chain2 <- Gibbsq2(iter=10000,X = X2,y=y,burnin=1000,tau_0=1,a=a,g=g,b0=b0,Kinv=Kinv)

# Remove every second iteration to reduce auto - corrlation

chain1t <- chain1[seq(1,dim(chain1)[1],by=2),]
chain2t <- chain2[seq(1,dim(chain2)[1],by=2),]
```

Reporting posterior means, standard deviations and 95 % central credible intervals for $\beta_0, \beta_1, \sigma^2$ by combining results for the two chains.

```
chain12t <- rbind(chain1t,chain2t)
chain_stats <- data.frame(matrix(nrow = 3,ncol =4 ))
para_names <- c('beta0','beta1','sigma2','lower_CI95','upper_CI95')
for (i in 1:3) {
  chain <-  chain12t[,i]
  quat <- quantile(sort(chain), c(0.05, 0.975))
  chain_stats[i,] <- c(mean(chain),sd(chain),quat)
}
names(chain_stats)<- c("posterior mean","std",'lower_CI95','upper_CI95')
```

```r
row.names(chain_stats) <- c('beta0','beta1','sigma2')

# chain results
chain_stats
```

```
##        posterior mean         std lower_CI95 upper_CI95
## beta0      4.12918718 0.111765425 3.94303272 4.35034827
## beta1      0.02963795 0.005814982 0.02024566 0.04106363
## sigma2     1.33588228 0.103746049 1.17458517 1.55165944
```