

# 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)
```

## Question Two (20 marks)

In lecture 3, we discussed how a Bayesian framework readily lends itself to combining information from sequential experiments. To demonstrate, consider the following data extracted from the *HealthIron* study.

Serum ferritin levels were measured for two samples of women, one of C282Y homozygotes ( $n = 88$ ) and the other of women with neither of the key mutations (C282Y and H63D) in the HFE gene, so-called HFE 'wildtypes' ( $n = 242$ ). The information available is

- **idnum**: Participant id.
- **homc282y**: Indicator whether individual is Homozygote (1) or Wildtype (0).
- **time**: Time since onset of menopause, measured in years.
- **logsf**: The natural logarithm of the serum ferritin in  $\mu\text{g/L}$ .

The data required to answer this question are `Hiron.csv`, which can be downloaded from LMS.

- a) Fit a standard linear regression,

$$E(\text{logsf}) = \beta_0 + \beta_1 \text{time}$$

with responses restricted to those who are homozygote ( $\text{homc282y} = 1$ ). This can be done using the `lm` function in R. Report the estimated coefficients  $\hat{\beta}$ , estimated error variance,  $\hat{\sigma}_e^2$  and  $(\mathbf{X}'\mathbf{X})^{-1}$ .

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

```

Hiron <- read.csv("Hiron.csv")
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

#Estimated error variance
sigma(model)^2

## [1] 1.715042

```

- b) Fit a Bayesian regression using a Gibbs sampler to **only the wildtype (homc282y=0) data**. Use the output from your answer in a) to define proper priors for  $\beta, \tau$ . For help, refer to lecture 13. For the Gibbs sampler, run two chains for 10,000 iterations. Discard the first 1000 iterations as burn-in and then remove every second remaining iteration to reduce auto-correlation. When storing results, convert  $\tau$  back to  $\sigma^2$ . When running the Gibbs sampler, incorporate posterior predictive checking, using the test statistic  $T(y, \beta) = \sum_{i=1}^n e_i^2$  and  $T(y^{\text{rep}}, \beta) = \sum_{i=1}^n (e_i^{\text{rep}})^2$ , where  $e_i$  is the predicted residual for observation  $i$  at simulation  $j$  and  $e_i^{\text{rep}}$  is the replicate residual for observation  $i$  at simulation  $j$ . Report posterior means, standard deviations and 95 % central credible intervals for  $\beta_0, \beta_1, \sigma^2$  combining results for the two chains.

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

$$p(\theta|\mathbf{y}) = \frac{p(\mathbf{y}_2|\theta)p(\theta|\mathbf{y}_1)}{p(\mathbf{y}_2|\mathbf{y}_1)}$$

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

$$p(\beta|\tau) = \mathcal{N}(\hat{\beta}_1, (\mathbf{X}_1\mathbf{X}_1)^{-1}/\tau)$$

$$p(\beta|\tau) = Ga\left(\frac{n_1 - p}{2}, \frac{(n_1 - p)s^2}{2}\right)$$

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  $\tau_\beta$ , and in all other places in the joint distribution, replace  $\tau_\beta$  and  $\tau_e$  with  $\tau$ .

- $\mathbf{K} = (\mathbf{X}_1\mathbf{X}_1)^{-1}$
- $\beta_0 = \hat{\beta}_1$
- $\alpha = \frac{n_1 - p}{2}$
- $\gamma = \frac{(n_1 - p)s^2}{2}$

Thus we get the following conditional posteriors

$$p(\tau|\mathbf{y}, \beta, \beta_0, \mathbf{K}) = \text{Ga}\left(\alpha + \frac{n+p}{2}, \gamma + \frac{(\mathbf{y} - \mathbf{X}\beta)'(\mathbf{y} - \mathbf{X}\beta) + (\beta - \beta_0)' \mathbf{K}^{-1} (\beta - \beta_0)}{2}\right)$$

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

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

```
# 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%*%b
    y_rep <- rnorm(n,mean = Xb,sd = 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 - correlation

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")

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

```

Performing posterior predictive checking we have a p value

```

chain12t_df <- data.frame(chain12t)
pval <- prop.table(table(chain12t_df$T_y>chain12t_df$T_rep))["TRUE"]
pval

```

```

##      TRUE
## 0.004777778

```

The p value seems to be very low so the model does not seem plausible.

- c) Perform convergence checks for the chain obtained in b). Report both graphical summaries and Gelman-Rubin diagnostic results. For the calculation of Gelman-Rubin diagnostics, you will need to install the R package coda. An example of processing chains for calculating Gelman-Rubin diagnostics is given below.

Processing chains for calculation of Gelman-Rubin diagnostics. Imagine you have 4 chains of a multi-parameter problem, and thinning already completed, called par1,par2,par3,par4

```

Step one: Converting the chains into mcmc lists.
library(coda)
par1<-as.mcmc.list(as.mcmc((par1)))
par2<-as.mcmc.list(as.mcmc((par2)))

```

```
par3<-as.mcmc.list(as.mcmc((par3)))
par4<-as.mcmc.list(as.mcmc((par4)))
```

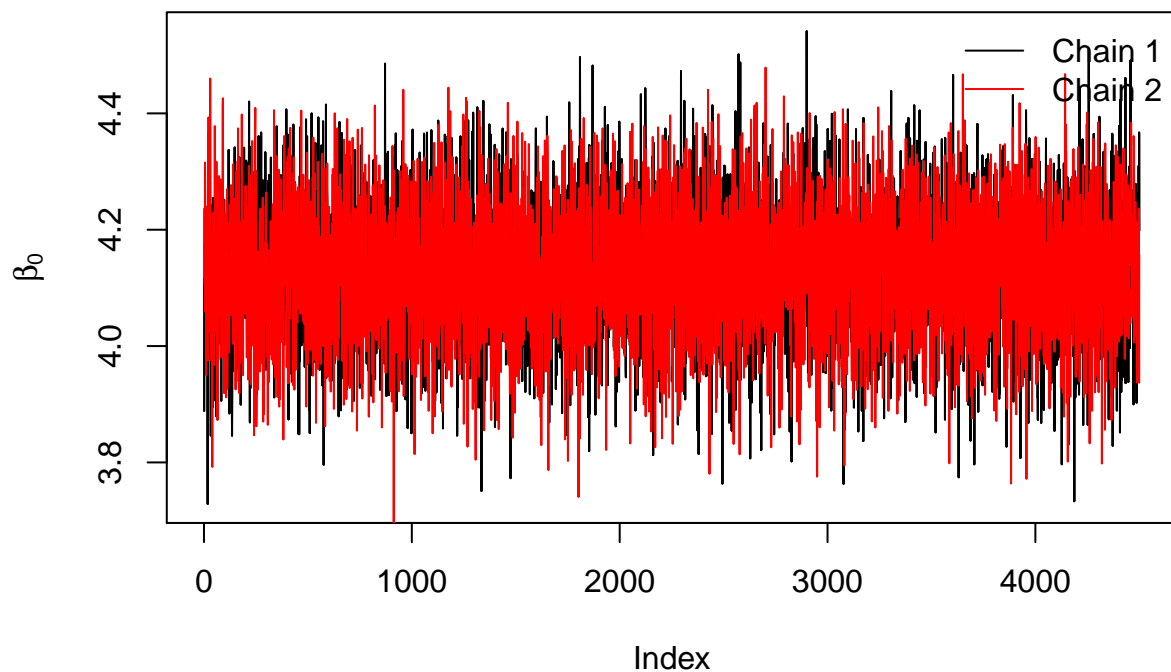
Step two: Calculating diagnostics

```
par.all<-c(par1,par2,par3,par4)
gelman.diag(par.all)
```

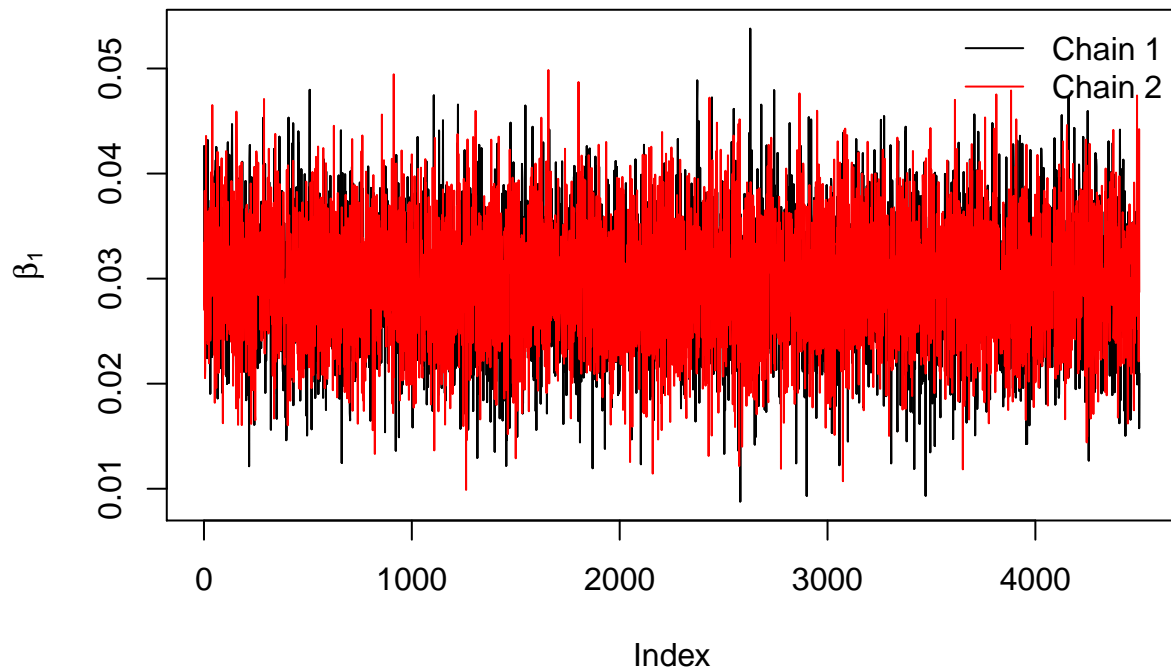
Answer:

PART C

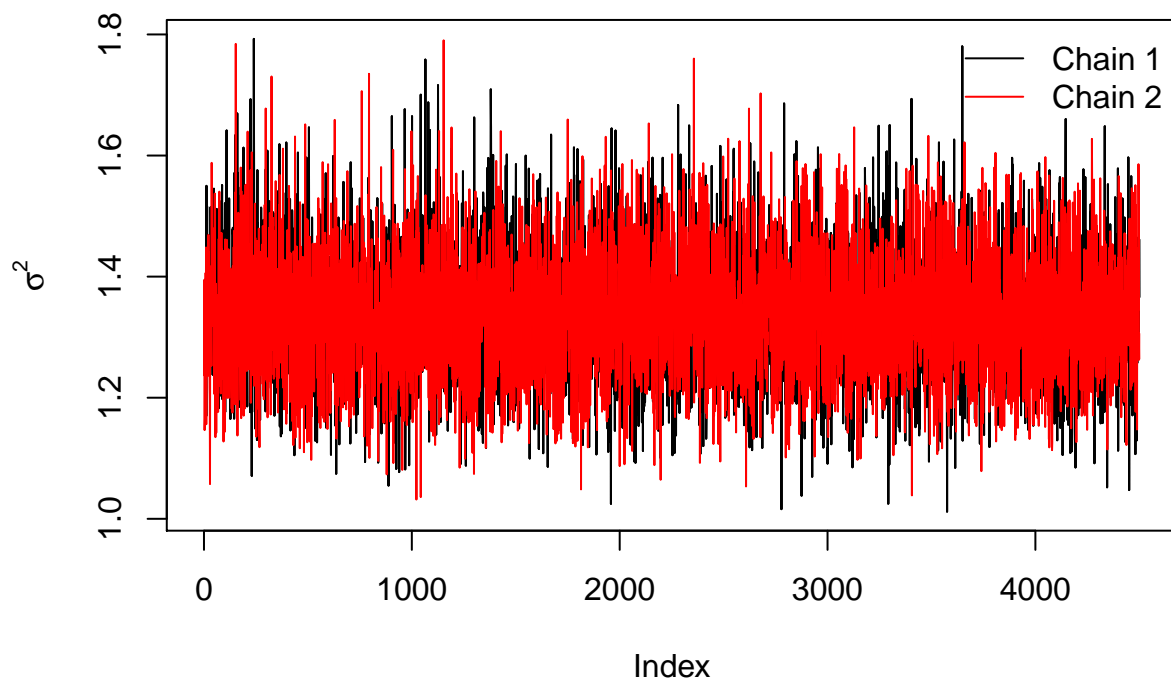
```
# beta0
plot(chain1t[,1],type='l',ylab=expression(beta[0]),col=1)
lines(chain1t[,1],type='l',col=1,ylab=expression(beta[0]))
lines(chain2t[,1],type='l',col=2,ylab=expression(beta[0]))
legend('topright',legend=c('Chain 1','Chain 2'),col=1:2,lty=1,bty='n')
```



```
# beta1
plot(chain1t[,2],type='l',ylab=expression(beta[1]))
lines(chain1t[,2],type='l',col=1,ylab=expression(beta[1]))
lines(chain2t[,2],type='l',col=2,ylab=expression(beta[1]))
legend('topright',legend=c('Chain 1','Chain 2'),col=1:2,lty=1,bty='n')
```



```
# sigma^2
plot(chain1t[,3],type='l',ylab=expression(sigma^2))
lines(chain1t[,3],type='l',col=1,ylab=expression(sigma^2))
lines(chain2t[,3],type='l',col=2,ylab=expression(sigma^2))
legend('topright',legend=c('Chain 1','Chain 2'),col=1:2,lty=1,bty='n')
```



```
m11<-as.mcmc.list(as.mcmc((chain1t[1:2250,])))
m12<-as.mcmc.list(as.mcmc((chain2t[1:2250,])))
m13<-as.mcmc.list(as.mcmc((chain1t[2250+1:2250,])))
m14<-as.mcmc.list(as.mcmc((chain2t[2250+1:2250,])))
estm1<-c(m11,m12,m13,m14)
```

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

```
##          Point est. Upper C.I.
## beta0    1.000018  1.0005800
## beta1    1.000009  1.0006422
## sigma2   1.000389  1.0013772
## T_y      0.999787  0.9998886
## T_rep    1.000444  1.0019675
```

The graphical summaries ensure that chain converge to the same distribution. The Gelman-Rubin diagnostics for all parameters are very close to one so all thus chains mixed sufficiently.

d) Fit a standard linear regression,

$$E(\text{logsf}) = \beta_0 + \beta_1 \text{time}$$

to **all the data** using the `lm` function in R. Report  $\hat{\beta}$ , and associated 95 % confidence intervals. Comparing these results to the results from b), do you believe that sequential analysis gave the same results as fitting the regression on the full data.

Answer:

```
# Fit regresssion for all of the data
```

```
full_model <- lm(logsf ~ time,data = Hiron)
summary(full_model)
```

```
##
## Call:
## lm(formula = logsf ~ time, data = Hiron)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.5197 -0.6597  0.0770  0.7523  2.6978
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.130162   0.110870  37.252  < 2e-16 ***
## time         0.029599   0.005769   5.131 4.95e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.152 on 328 degrees of freedom
## Multiple R-squared:  0.07429,    Adjusted R-squared:  0.07147
## F-statistic: 26.32 on 1 and 328 DF,  p-value: 4.952e-07
```

```
# Estimates
```

```
full_model$coefficients
```

```
## (Intercept)      time
##  4.13016198  0.02959941
```

```
# 95 % confidence intervals
```

```
confint(full_model)
```

```
##              2.5 %      97.5 %
```

```
## (Intercept) 3.91205524 4.34826873
## time        0.01825025 0.04094857
```

Comparing the results from the full linear model (the cell above) and the sequential model from part b (chain\_stats) we conclude that both models achieved the same results (up to two decimal places).

- e) Report the results of posterior predictive checking requested in b). Do you believe the postulated model was plausible. If not, what do you think is a potential flaw in the postulated model.