Assignment 5

Mayank Raunak

December 3, 2019

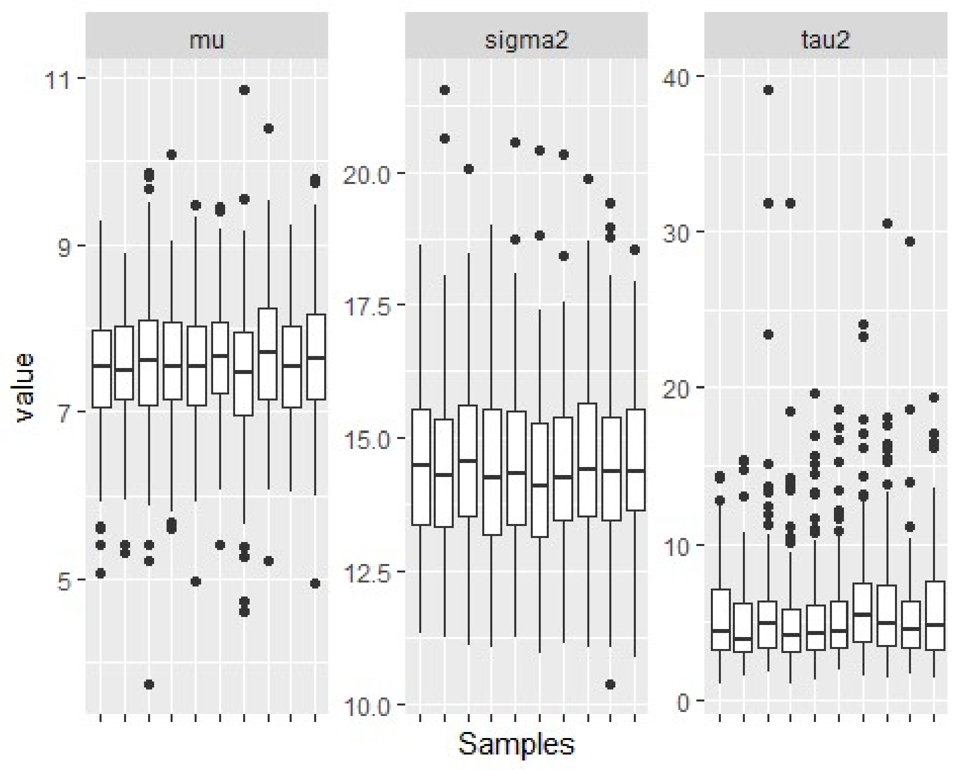
8.3

a

```
# Load data
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(tidyr)
library(MCMCpack)

## Loading required package: coda

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)

## ## Copyright (C) 2003-2019 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park

## ##
## ## Support provided by the U.S. National Science Foundation

## ## (Grants SES-0350646 and SES-0350613)
## ##

data_school = lapply(1:8, function(i) {school.table = paste0('http://www.stat.washing
ton.edu/people/pdhoff/Book/Data/hwdata/school', i, '.dat') %>%url %>%read.table

  data.frame(school = i,hours = school.table[, 1] %>% as.numeric)
})
```

```r
Y = do.call(rbind, data_school)

# Reference was taken from the text book
# Informative Prior
mu0<- 7; g20<-5
t20<-10; eta0<-2
s20<-15; nu0 = 2

## Starting values
# let number of schools be m, the fiest column of Y matrix is school id, hence length
will give number of schools
m<- length(unique(Y[, 1]))
n<-sv<-ybar<-rep(NA, m)

for (j in 1:m) {
  Y_j = Y[Y[, 1] == j, 2]
  ybar[j] = mean(Y_j)
  sv[j] = var(Y_j)
  n[j] = length(Y_j)
}

# Let initial theta estimates be the sample means and initial values of sigma2, mu, a
nd tau2 be "sample mean and variance"
theta<-ybar ; sigma2<-mean( sv )
mu<-mean( theta ) ; tau2<-var ( theta )

### setup MCMC
set.seed (1)
S<-1500
THETA<-matrix ( nrow=S, ncol=m)
SMT<-matrix ( nrow=S, ncol=3)
colnames(SMT) = c('sigma2', 'mu', 'tau2')

### MCMC algorithm
for (s in 1:S) {
  # Sample new values of the thetas
  for (j in 1:m) {
    vtheta = 1 / (n[j] / sigma2 + 1 / tau2)
    etheta = vtheta * (ybar[j] * n[j] / sigma2 + mu / tau2)
    theta[j] = rnorm(1, etheta, sqrt(vtheta))
  }

  # Sample new values of sigma2
  nun<-nu0+sum(n)
  ss<-nu0*s20
  for ( j in 1:m){ss<-ss+sum((Y[Y[,1]==j ,2]-theta [ j ])^2)}
  sigma2<-1/rgamma(1 ,nun/2 , ss /2)


  #sample a new value of mu
  vmu<- 1/(m/tau2+1/g20)
  emu<- vmu*(m*mean( theta )/tau2 + mu0/g20)
  mu<-rnorm(1 ,emu, sqrt (vmu))
  # sample a new value of tau2
  etam = eta0 + m
  ss = eta0 * t20 + sum((theta - mu)^2)
  tau2 = 1 / rgamma(1, etam / 2, ss / 2)

  #store results
  THETA[s, ] = theta
  SMT[s, ] = c(sigma2, mu, tau2)
}
```

Assess convergence with diagnostic boxplots:



Evaluate effective sample size:

```
# Run the chain long enough so that the effective sample sizes are all above 1,000
library(coda)
effectiveSize(SMT[, 1])

## var1
## 1500

effectiveSize(SMT[, 2])

##     var1
## 1252.463

effectiveSize(SMT[, 3])

##     var1
## 1091.732
```

b

Compute posterior means and 95% confidence regions for {sigma2,mu,tau2}. Also,compared the posterior densities to the prior densities.

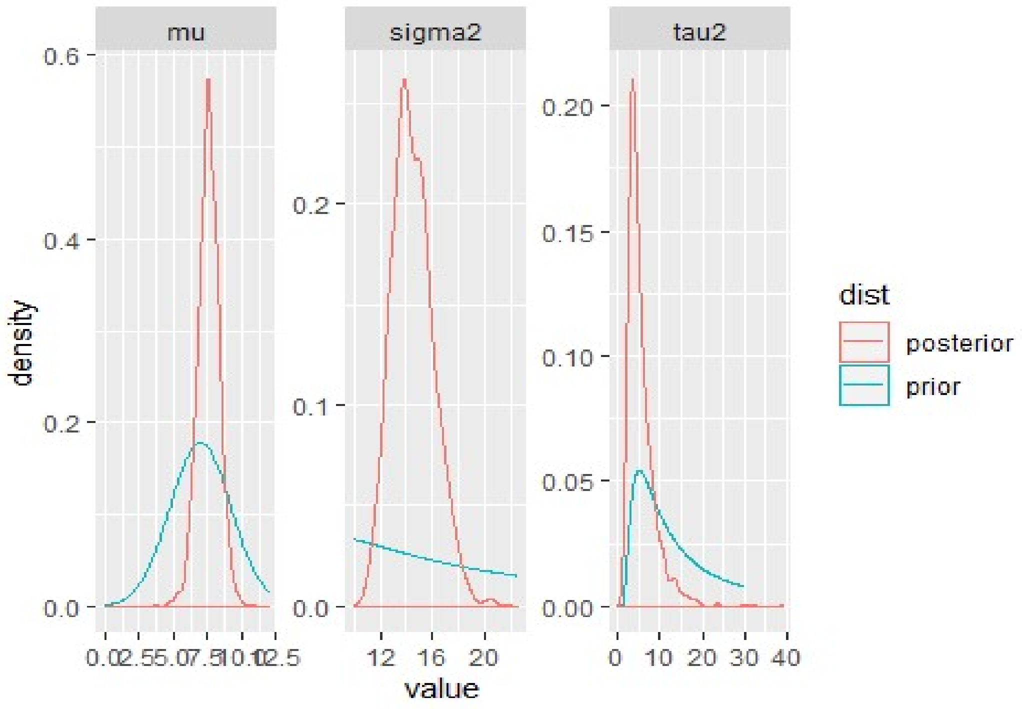
Posterior means and 95% confidence intervals

```
t(apply(SMT, MARGIN = 2, FUN = quantile, probs = c(0.025, 0.5, 0.975)))
```

```
##              2.5%       50%     97.5%
## sigma2 11.736137 14.321127 17.843343
## mu      6.058533  7.567649  9.054348
## tau2    1.878153  4.474879 14.341699
```

comparison between the posterior densities to the prior densities

```
prior.sigma2 = data.frame(value = seq(10, 22.5, by = 0.1),density = dinvgamma(seq(10, 22.5, by = 0.1), nu0 / 2, nu0 * s20 / 2),variable = 'sigma2')
prior.tau2 = data.frame(value = seq(0, 30, by = 0.1),density = dinvgamma(seq(0, 30, by = 0.1), eta0 / 2, eta0 * t20 / 2),variable = 'tau2')

prior.mu = data.frame(value = seq(0, 12, by = 0.1),density = dnorm(seq(0, 12, by = 0.1), mu0, sqrt(g20)),variable ='mu')

dataframe_priors = rbind(prior.sigma2, prior.tau2, prior.mu)
dataframe_priors$dist = 'prior'
dataframe_SMT$dist = 'posterior'
ggplot(dataframe_priors, aes(x = value, y = density, color = dist)) +geom_line() +geom_density(data = dataframe_SMT, mapping = aes(x = value, y = ..density..)) +facet_wrap(~ variable, scales = 'free')
```

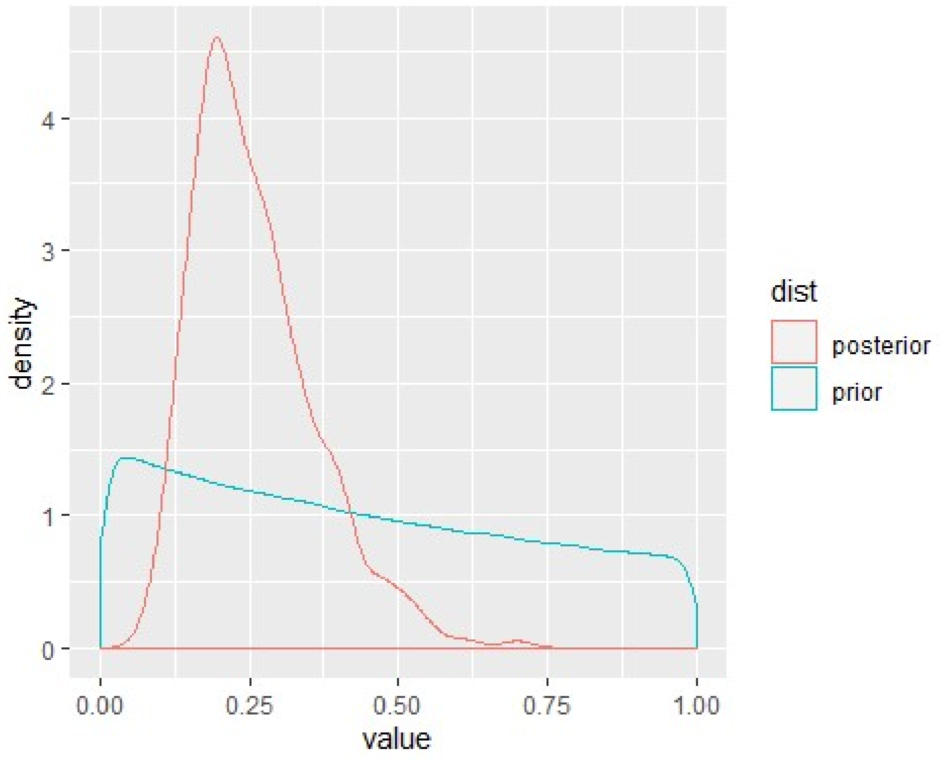


We can conclude that the prior estimates of mu and tau2 are accurate, but the prior estimates for sigma2 is not accurate.

c

Plot the posterior density of R = τ2/σ2+τ2 and compare it to a plot of the prior density of R. Describe the evidence for between-school variation

```
prior_t20= (1 / rgamma(1e6, eta0 / 2, eta0 * t20 / 2))
prior_s20 = (1 / rgamma(1e6, nu0 / 2, nu0 * s20 / 2))
prior_R = data.frame(value = (prior_t20) / (prior_t20 + prior_s20),dist = 'prior')

posterior_R = data.frame(value = SMT[, 'tau2'] / (SMT[, 'tau2'] + SMT[, 'sigma2']),di
st = 'posterior')
ggplot(prior_R, aes(x = value, y = ..density.., color = dist)) +
  geom_density(data = prior_R) +geom_density(data = posterior_R)
```



```
mean(posterior_R$value)

## [1] 0.2588751
```

R is used to measure how much between-school variance is existing. The priors didn't encode this variable, but post inference, we can conclude that there is around 25% variance from between-school (tau2)

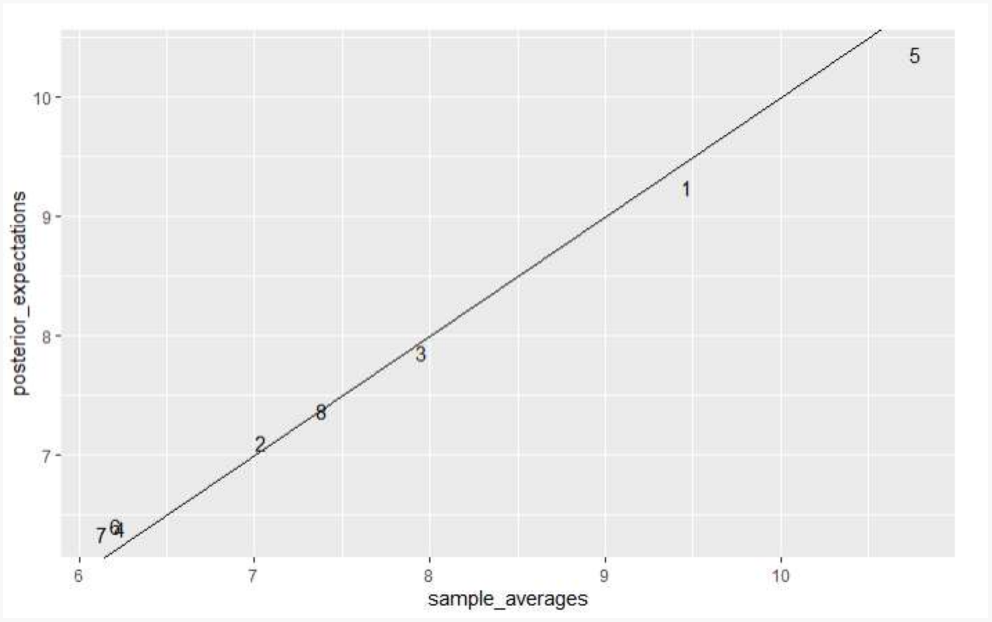
Obtain the posterior probability that θ7 is smaller than θ6, as well as the posterior probability that θ7 is the smallest of all the θ's.

```
#posterior probability that θ7 is smaller than θ6
smaller = THETA[, 7] < THETA[, 6]
mean(smaller)

## [1] 0.53

#posterior probability that θ7 is the smallest of all the θ's.
smallest = (THETA[, 7] < THETA[, -7]) %>%apply(MARGIN = 1, FUN = all)
mean(smallest)

## [1] 0.3146667
```

E

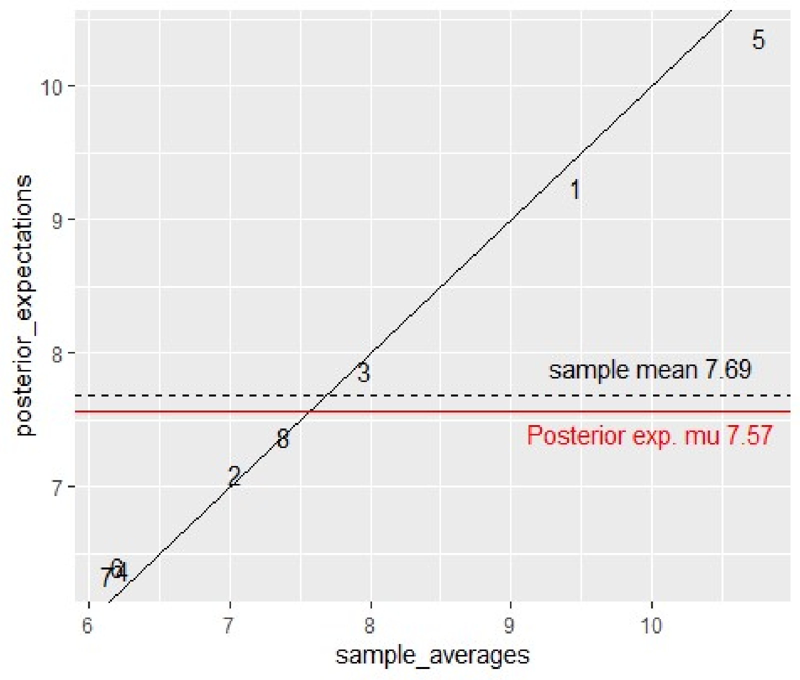
Plot of sample averages(ybar) against posterior expectation of thetas and computed the sample mean of all the observations and compared it to mean of mu.

```
relation = data.frame(sample_averages = ybar,posterior_expectations = colMeans(THETA)
,school = 1:length(ybar))
ggplot(relation, aes(x = sample_averages, y = posterior_expectations, label = school)
) +
  geom_text() +geom_abline(slope = 1, intercept = 0)
```



~~~
```r}
~~~

#computed the sample mean of all the observations and compared it to mean of mu.

```
print(mean(SMT[, 'mu']))

mean(Y[, 'hours'])

```
```

```
[1] 7.569173
[1] 7.691278
```

```
Plotting all the information in one graph for better visualization
```



5
1
3
sample mean 7.69
8
Posterior exp. mu 7.57
2
7
6
4
10
9
8
7
6
7
8
9
10
posterior_expectations
sample_averages

We can observe a very strong relationship between the sample average and the posterior expectation. However, it is a little off at the extremities.

9.1

```
swim = read.table(url('http://www.stat.washington.edu/people/pdhoff/Book/Data/hwdata/swim.dat'))
```

According to the given question, to formulate prior, we need to use the information that competitive times for this age group generally range from 22 to 24 seconds and hence , I chose prior time taken is 23 seconds, y-intercept to be 23, and we let the prior expectation of the effect of training week by week to be 0, so $\boldsymbol{\beta} = (23,0)^T$.

```
library(stats)
library(MASS)
library(dplyr)
```
```

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

## The following object is masked from 'package:MASS':
##
##     select

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

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

#Reference was taken from text book example
#Number of Independent samples to generate: S
S = 5000
X = cbind(rep(1, 6), seq(1, 11, by = 2))
n = dim(X)[1]
p = dim(X)[2]

# Prior parameters
beta0 = c(23, 0)
sigma0 = rbind(c(0.25, 0), c(0, 0.1))
nu0 = 1;s20 = 0.25
set.seed(1)

# I ran linear regression gibbs sampling and got the posterior.
# predictive distribution
swim_pred = apply(swim, MARGIN = 1, function(y) {

 # Store samples
 BETA = matrix(nrow = S, ncol = length(beta0))
 SIGMA = numeric(S)

 # Starting values
 beta = c(23, 0)
 s2 = 0.7^2

 # Gibbs sampling algorithm
 for (s in 1:S) {
  # Calculated V and m
  V = solve(solve(sigma0) + (t(X) %*% X) / s2)
  m = V %*% (solve(sigma0) %*% beta0 + (t(X) %*% y) / s2)

  #sampled one beta
  beta = mvrnorm(1, m, V)

  #Compute SSR(beta)
  ssr_g = (t(y) %*% y) - (2 * t(beta) %*% t(X) %*% y) + (t(beta) %*% t(X) %*% X %*% beta)

  # sample s2
  s2 = 1 / rgamma(1, (nu0 + n) / 2, (nu0 * s20 + ssr_g) / 2)
  BETA[s, ] = beta
  SIGMA[s] = s2
 }

 # Now sampling posterior predictive - two weeks later
 xpred = c(1, 13)
 YPRED = rnorm(S, BETA %*% xpred, sqrt(SIGMA))

 YPRED
})
```

b

```
fastest_times = apply(swim_pred, MARGIN = 1, FUN = which.min)
table(fastest_times) / length(fastest_times)

## fastest_times
##      1      2      3      4
## 0.6524 0.0134 0.3060 0.0282
```

We observed that swimmer 1 has improved the most as compared to others from posterior predictive dataset, so I would recommend Coach to select swimmer 1 for the swimming meet, wich will happend in two weeks.

9.3

```
crime_data = read.table(url('http://www.stat.washington.edu/people/pdhoff/Book/Data/hwdata/crime.dat'), hea
der = TRUE)
```

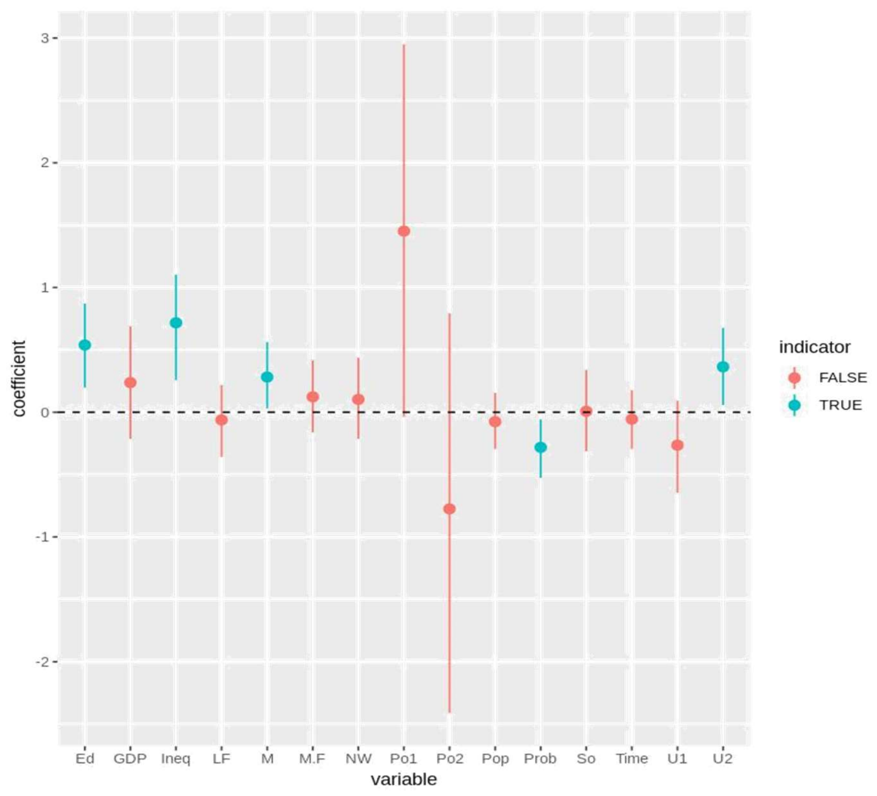
a

```
## Reference was taken from th text book
## data : y , X
## prior parameters : g , nu0 , s20
## number of independent samples to generate : S
## Number of explantory variables =15

y = crime_data$y
X = crime_data %>% select(-y) %>% as.matrix
n = dim(X)[1];p = dim(X)[2];g <-length(y);nu0 = 2;s20 = 1;S = 1000

Hg <- (g / (g + 1)) * X %*% solve(t(X) %*% X) %*% t(X)
SSRg <- t(y) %*% (diag(1, nrow = n) - Hg) %*% y

s2 <- 1 / rgamma(S, (nu0 + n) / 2, (nu0 * s20 + SSRg) / 2)

Vb <- g * solve(t(X) %*% X) / (g + 1)
Eb <- Vb %*% t(X) %*% y

E <-matrix(rnorm(S * p, 0, sqrt(s2)), S, p)
beta<-t(t(E %*% chol(Vb)) + c(Eb))
colMeans(beta)

##           M          So          Ed         Po1         Po2          LF
##  0.283217431  0.001327702  0.532312545  1.423871255 -0.747277546 -0.060002645
##         M.F         Pop          NW          U1          U2         GDP
##  0.128670887 -0.072466571  0.098612286 -0.271963727  0.371286474  0.238010041
##        Ineq        Prob        Time
##  0.715552927 -0.271707095 -0.053583276

## Coefficients obtains from ordinary least sqaure estimation
beta_ols = solve(t(X) %*% X) %*% t(X) %*% y
beta_ols

##              [,1]
## M     0.2865177028
## So   -0.0001179958
## Ed    0.5445161778
## Po1   1.4716146465
## Po2  -0.7817757455
## LF   -0.0659672893
## M.F   0.1313002714
## Pop  -0.0702910179
## NW    0.1090590127
## U1   -0.2705407273
## U2    0.3687335028
## GDP   0.2380580097
## Ineq  0.7262918200
## Prob -0.2852262729
## Time -0.0615771841
```

According to the ols estimates Ineq-income inequality with weight 0.92060778, Po2 with weight 0.760097, ED – mean years of schooling with weight 0.6 seem strongly predictive of the crime rates.

Now, Visualizing the marginal posterior means and 95% confidence intervals for β, and comparison to the least squares estimates and relationships between crime and the explanatory variables

```
library(ggplot2)
library(tidyr)
Important = apply(beta, MARGIN = 2, FUN = quantile, probs = c(0.025, 0.5, 0.975)) %>% apply(MARGIN = 2, FUN = f
unction(y) !(y[1] < 0 && 0 < y[3]))
beta_df = as.data.frame(beta) %>% gather(key = 'variable', val = 'coefficient') %>% mutate(Important = Important[
variable])


ggplot(beta_df, aes(x = variable, y = coefficient, color = Important)) + stat_summary(fun.y = mean, fun.ymin = func
tion(y) quantile(y, probs = c(0.025)), fun.ymax = function(y) quantile(y, probs = c(0.975))) +  geom_hline(yintercep
t = 0, lty = 2)
```



b

```
y = crime_data$y
X = crime_data %>% select(-y) %>% as.matrix
set.seed(50)

#Randomly dividing the crime data into in half-- traing set and test set
train_data = sample.int(length(y), size = round(length(y) / 2), replace = FALSE)

#train_set
train_y = y[train_data]
train_x = X[train_data, ]

#test_set
test_y = y[-train_data]
test_x = X[-train_data, ]
```

i

```
library(ggplot2)
## Coefficients obtains from ordinary least sqaure estimation
beta_ols = solve(t(train_x) %*% train_x) %*% t(train_x) %*% train_y
beta_ols

##             [,1]
## M     0.72778002
## So   -0.55360874
## Ed    0.34213419
## Po1   1.79517847
## Po2  -1.52499737
## LF   -0.10662966
## M.F   0.14722658
## Pop  -0.01820537
## NW    0.24063498
## U1   -0.12833679
## U2    0.34520010
## GDP   1.35199132
## Ineq  1.27778853
## Prob -0.22261862
## Time -0.02249125
```

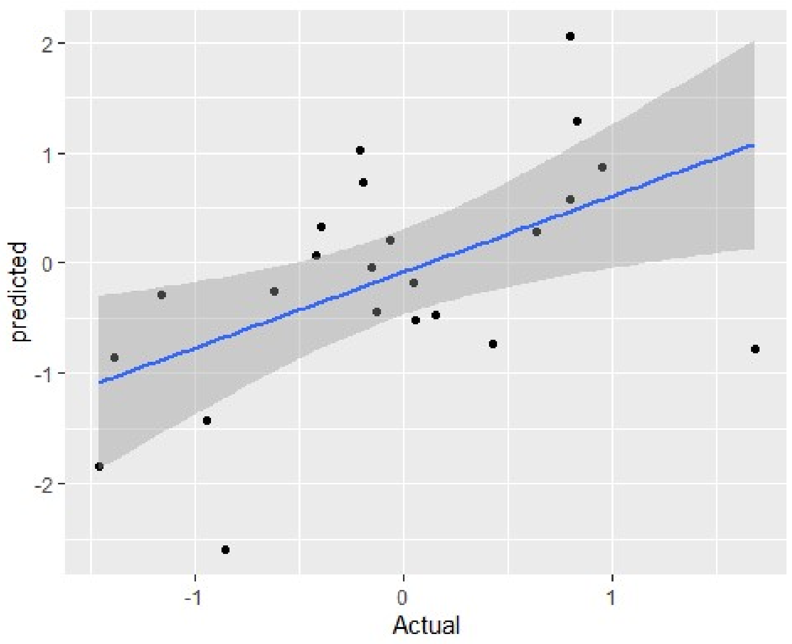
```
y_predicted = test_x %*% beta_ols



##Computing the prediction error for ols
pred_error = sum((test_y - y_predicted)^2) / length(test_y)
pred_error
```

```
## [1] 0.8446485
```

B ii

```
y = train_y
X = train_x
n = dim(X)[1]
p = dim(X)[2]
g = n;nu0 = 2;s20 = 1
S = 1000

Hg = (g / (g + 1)) * X %*% solve(t(X) %*% X) %*% t(X)
SSRg = t(y) %*% (diag(1, nrow = n) - Hg) %*% y
s2 = 1 / rgamma(S, (nu0 + n) / 2, (nu0 * s20 + SSRg) / 2)
Vb = g * solve(t(X) %*% X) / (g + 1)
Eb = Vb %*% t(X) %*% y
E = matrix(rnorm(S * p, 0, sqrt(s2)), S, p)
beta = t(t(E %*% chol(Vb)) + c(Eb))
beta_bayes = as.matrix(colMeans(beta))
y_bayes = test_x %*% beta_bayes
dataframe_bayes = data.frame(Actual = test_y,predicted = y_bayes)
ggplot(dataframe_bayes, aes(x = Actual, y = predicted)) +geom_point() +geom_smooth(method = 'lm')
```

Below is the plot between Actual target value and the predicted target value for Bayesian linear Regression



```
# Computation of the prediction error
prediction_error = sum((test_y - y_bayes)^2) / length(test_y)
prediction_error

## [1] 0.7841516
```

when the seed is 1, ther is minor difference between the prediction errors. But when seed value is not set, then there is significant difference between both the prediction errors

OLS prediction error for seed 1= 0.5474409 ; bayes prediction error for seed 1= 0.5063905

OLS prediction error by setting seed value to 50= 0.8446485

Bayes prediction error by setting seed value =0.7841516

We can conclude that, bayes prediction error is less as comapred to the OLS prediction error in general case, as it also takes into consideration the prior information and Bayesian Linear Regresssion aims to find the posterior distribution for the model parameters rather than finding the single best value.

References:

A First Course in Bayesian Statistics - Peter Hoff 2009 Class Notes- R code

https://www.tutorialspoint.com/r/index.htm,

https://www.tutorialspoint.com/r ,http://www.r-tutor.com/ ,

http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html

http://rstudio-pubsstatic.s3.amazonaws.com/4952_7af63ededf804cbc9dbc85dda50d07e3.html

https://github.com/jayelm/hoff-bayesian-statistics

https://stackoverflow.com/questions/2871763/