DASC6510 Assignment2

Hao Xu T00732492

Source code of this document: https://github.com/cld4h/DASC_6510_Assignment/tree/master/AST2

Problem 1: Exercise 8.3 (page 242)

Hierarchical modeling: The files school1.dat through school8.dat give weekly hours spent on homework for students sampled from eight different schools. Obtain posterior distributions for the true means for the eight different schools using a hierarchical normal model with the following prior parameters: (Refer to Page 132)

$$\mu_0 = 7, \gamma_0^2 = 5, \quad \tau_0^2 = 10, \eta_0 = 2, \quad \sigma_0^2 = 15, \nu_0 = 2.$$

Part a

a) Run a Gibbs sampling algorithm to approximate the posterior distribution of $\{\boldsymbol{\theta}, \sigma^2, \mu, \tau^2\}$. Assess the convergence of the Markov chain, and find the effective sample size for $\{\sigma^2, \mu, \tau^2\}$. Run the chain long enough so that the effective sample sizes are all above 1,000.

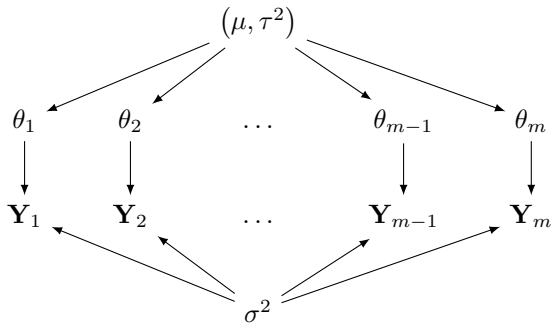


Figure 1: Graphical representation of the hierarchical model.

Because here only $\boldsymbol{\theta}$ is a vector ($\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)^T$, $m = 8$ is the total number of schools), σ^2 is the same across all 8 schools, so we are assuming common variance. The fixed but unknown parameters in this model are μ, τ^2 and σ^2. We use standard semiconjugate normal and inverse-gamma prior distributions for these parameters:

$$\frac{1}{\sigma^2} \sim \text{gamma}(\frac{\nu_0}{2}, \frac{\nu_0\sigma_0^2}{2})$$
$$\frac{1}{\tau^2} \sim \text{gamma}(\frac{\eta_0}{2}, \frac{\eta_0\tau_0^2}{2})$$
$$\mu \sim \text{normal}(\mu_0, \gamma_0^2)$$

Posterior approximation proceeds by iterative sampling of each unknown quantity from its full conditional distribution. Given a current state of the unknowns $\{\theta_1^{(s)}, \ldots, \theta_m^{(s)}, \mu^{(s)}, \tau^{2(s)}, \sigma^{2(s)}\}$, a new state is generated as follows:

1. sample $\mu^{(s+1)} \sim p(\mu|\theta_1^{(s)}, \ldots, \theta_m^{(s)}, \tau^{2(s)})$
2. sample $\tau^{2(s+1)} \sim p(\tau^2|\theta_1^{(s)}, \ldots, \theta_m^{(s)}, \mu^{(s+1)})$
3. sample $\sigma^{2(s+1)} \sim p(\sigma^2|\theta_1^{(s)}, \ldots, \theta_m^{(s)}, \mathbf{Y}_1, \ldots, \mathbf{Y}_m)$
4. for each $j \in \{1, \ldots, m\}$, sample $\theta_j^{(s+1)} \sim p(\theta_j|\mu^{(s+1)}, \tau^{2(s+1)}, \sigma^{2(s+1)}, \mathbf{Y}_j)$

$$\{\mu|\theta_1, \ldots, \theta_m, \tau^2\} \sim \text{normal}\left(\frac{m\bar{\theta}/\tau^2 + \mu_0/\gamma_0^2}{m/\tau^2 + 1/\gamma_0^2}, [m/\tau^2 + 1/\gamma_0^2]^{-1}\right)$$
$$\{\frac{1}{\tau^2}|\theta_1, \ldots, \theta_m, \mu\} \sim \text{gamma}\left(\frac{\eta_0 + m}{2}, \frac{\eta_0\tau_0^2 + \sum(\theta_j - \mu)^2}{2}\right)$$
$$\{\theta_j|Y_{1,j}, \ldots, Y_{n_j,j}, \sigma^2\} \sim \text{normal}\left(\frac{n_j\overline{Y_{\cdot,j}}/\sigma^2 + 1/\tau^2}{n_j/\sigma^2 + 1/\tau^2}, [n_j/\sigma^2 + 1/\tau^2]^{-1}\right)$$
$$\{\frac{1}{\sigma^2}|\theta_1, \ldots, \theta_m, Y_1, \ldots, Y_n\} \sim \text{gamma}\left(\frac{1}{2}\left[\nu_0 + \sum_{j=1}^{m} n_j\right], \frac{1}{2}\left[\nu_0\sigma_0^2 + \sum_{j=1}^{m}\sum_{i=1}^{n_j}(Y_{i,j} - \theta_j)^2\right]\right)$$

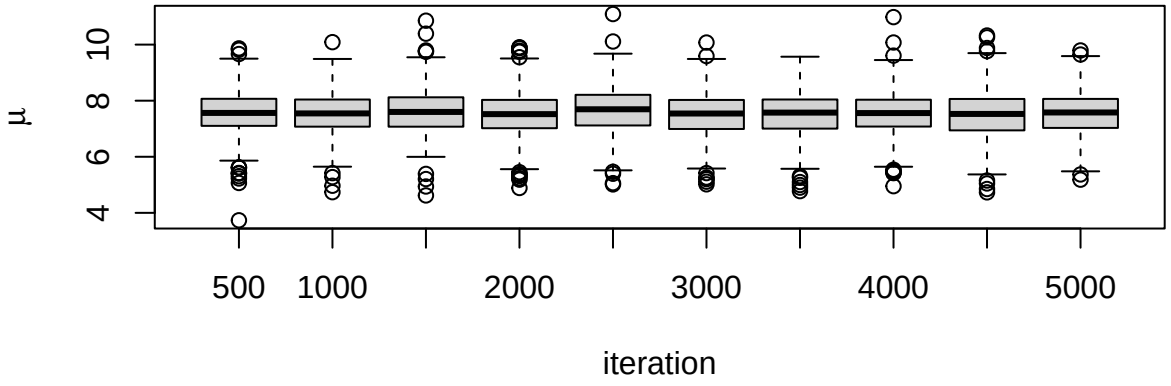
```
### Code modified from Ch.8.4.1
### weakly informative priors
mu0 <- 7; g20 <- 5
t20 <- 10; eta0 <- 2
s20 <- 15; nu0 <- 2
###

Y <- NULL
### load the data
# iterate from 1 to 8
for (i in 1:8) {
# store schooli.dat into Y
scores <- scan(paste0("school",i,".dat"))
      for(j in 1:length(scores)){
        Y <- rbind(Y, c(i,scores[j]))
      }
}

## starting values
m<-length(unique(Y[,1])) # the number of schools, m=8
n<-sv<-ybar<-rep(NA,m)
for(j in 1:m)
```

```
{
  # mean score for each school
  ybar[j]<-mean(Y[Y[,1]==j,2])
  # score variance for each school
  sv[j]<-var(Y[Y[,1]==j,2])
  # number of students in each school
  n[j]<-sum(Y[,1]==j)
}
# initial value?
theta<-ybar; sigma2<-mean(sv)
mu<-mean(theta); tau2<-var(theta)
###

## setup MCMC
set.seed(1)
S<-5000
THETA<-matrix( nrow=S,ncol=m)
MST<-matrix( nrow=S,ncol=3)

## MCMC algorithm
for(s in 1:S)
{

  # sample new values of the thetas
  for(j in 1:m)
  {
    # Variance of full conditional distribution of
    # theta given y_{1,j}..y_{n_j,j} and \sigma^2
    vtheta<-1/(n[j]/sigma2+1/tau2)
    # Mean of full conditional distribution of
    # theta given y_{1,j}..y_{n_j,j} and \sigma^2
    etheta<-vtheta*(ybar[j]*n[j]/sigma2+mu/tau2)
    theta[j]<-rnorm(1,etheta,sqrt(vtheta))
   }

  # sample new value of sigma2 from inverse-gamma
  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 from normal
  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 from gamma
  etam<-eta0+m
  ss<- eta0*t20 + sum( (theta-mu)^2 )
  tau2<-1/rgamma(1,etam/2,ss/2)

  #store results
```

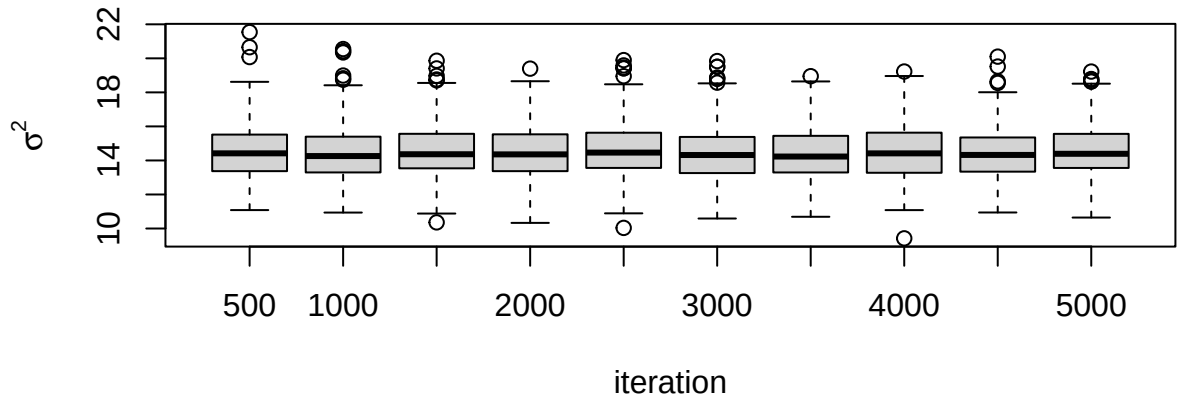
```
  THETA[s,]<-theta
  MST[s,]<-c(mu,sigma2,tau2)
}
```

Assess the convergence of the Markov chain

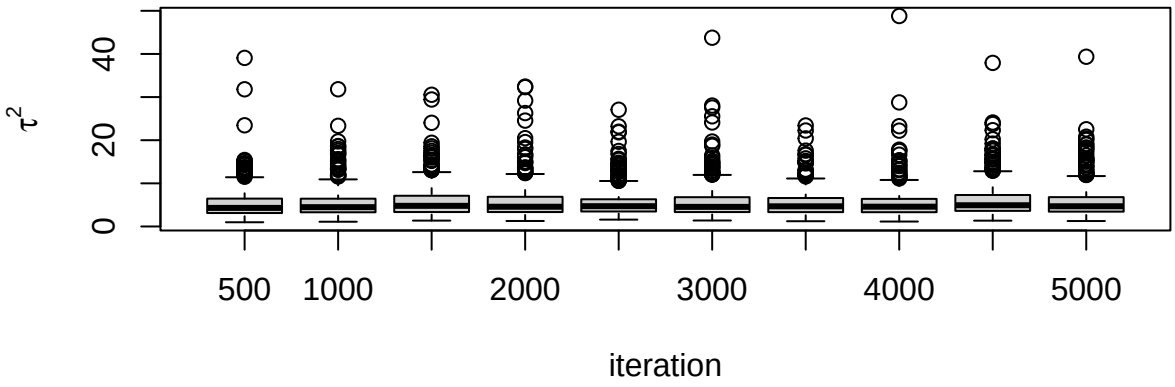
```
#### A plot for evaluating lack of convergence
stationarity.plot<-function(x,...){

S<-length(x)
scan<-1:S
ng<-min( round(S/100),10)
group<-S*ceiling( ng*scan/S) /ng

boxplot(x~group,...)
}
stationarity.plot(MST[,1],xlab="iteration",ylab=expression(mu))
```




```
stationarity.plot(MST[,2],xlab="iteration",ylab=expression(sigma^2))
```




```
stationarity.plot(MST[,3],xlab="iteration",ylab=expression(tau^2))
```



Find the effective sample size for $\{\mu, \sigma^2, \tau^2\}$.

```
library(coda)
# The effective size for mu is:
effectiveSize(MST[, 1])
```

```
##     var1
## 4177.568
```

```
# The effective size for sigma^2 is:
effectiveSize(MST[, 2])
```

```
##     var1
## 4739.021
```

```
# The effective size for tau^2 is:
effectiveSize(MST[, 3])
```

```
##     var1
## 3618.343
```

Part b

b) Compute posterior means and 95% confidence regions for $\{\sigma^2, \mu, \tau^2\}$. Also, compare the posterior densities to the prior densities, and discuss what was learned from the data.

```
B <- 201 # Burn-in first 200 samples
library(xtable)

Conf.Regions <- t(apply(MST[B:S,], MARGIN = 2, FUN = quantile, probs = c(0.025, 0.5, 0.975)))
Post.Mean <- t(t(apply(MST[B:S,], MARGIN = 2, FUN = mean)))
cbind(Post.Mean,Conf.Regions)
```

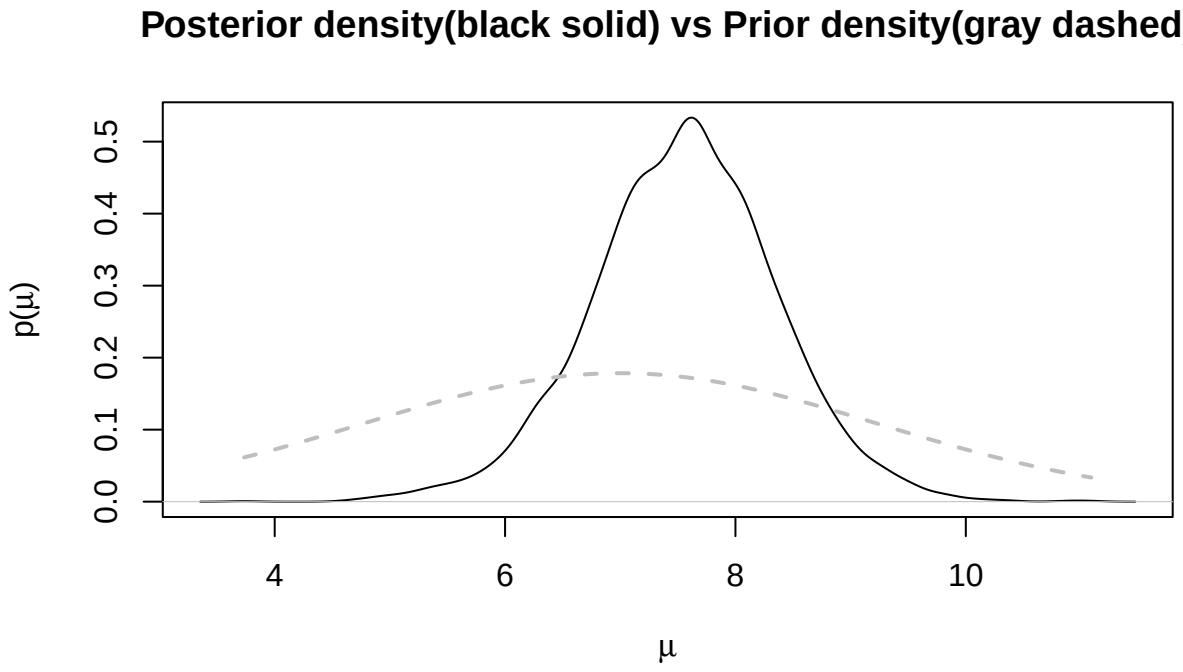
```
##                     2.5%       50%    97.5%
## [1,]  7.551726  5.917359  7.569239  9.13198
## [2,] 14.479088 11.731138 14.337219 17.79038
## [3,]  5.610614  1.899229  4.627957 14.97397
```

```
# xtable( cbind(Post.Mean,Conf.Regions))
```

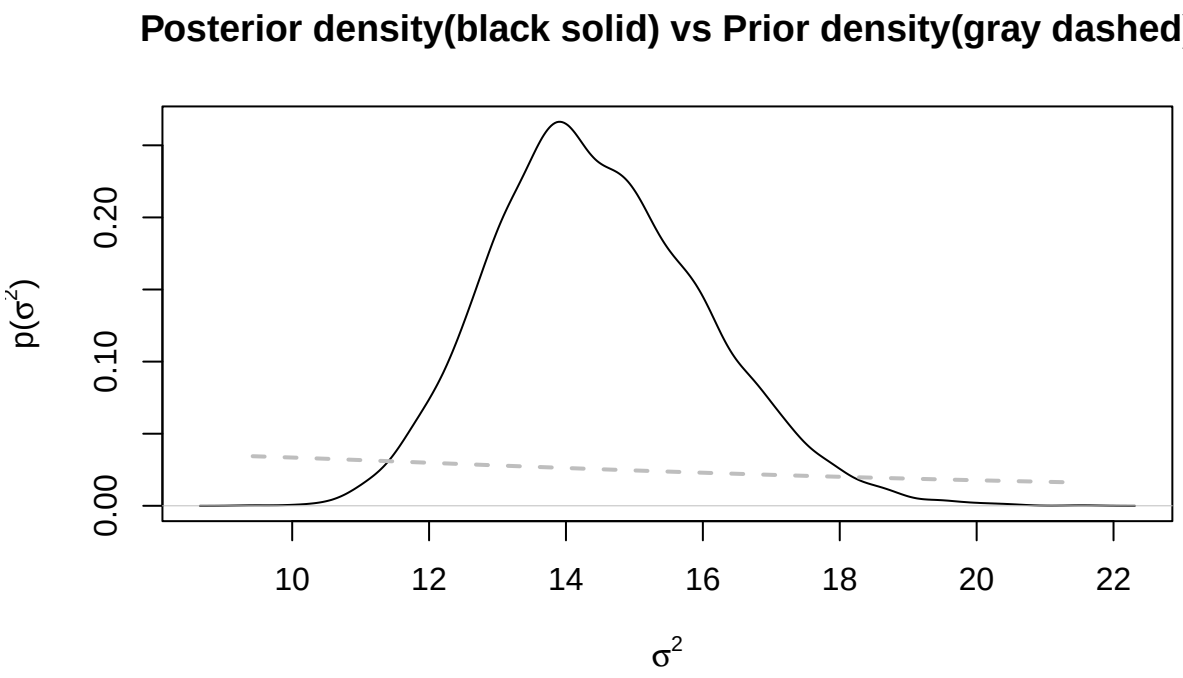
	Posterior Mean	2.5%	50%	97.5%
μ	7.55	5.92	7.57	9.13
σ^2	14.48	11.73	14.34	17.79
τ^2	5.61	1.90	4.63	14.97

Comparing posterior densities to the prior densities:

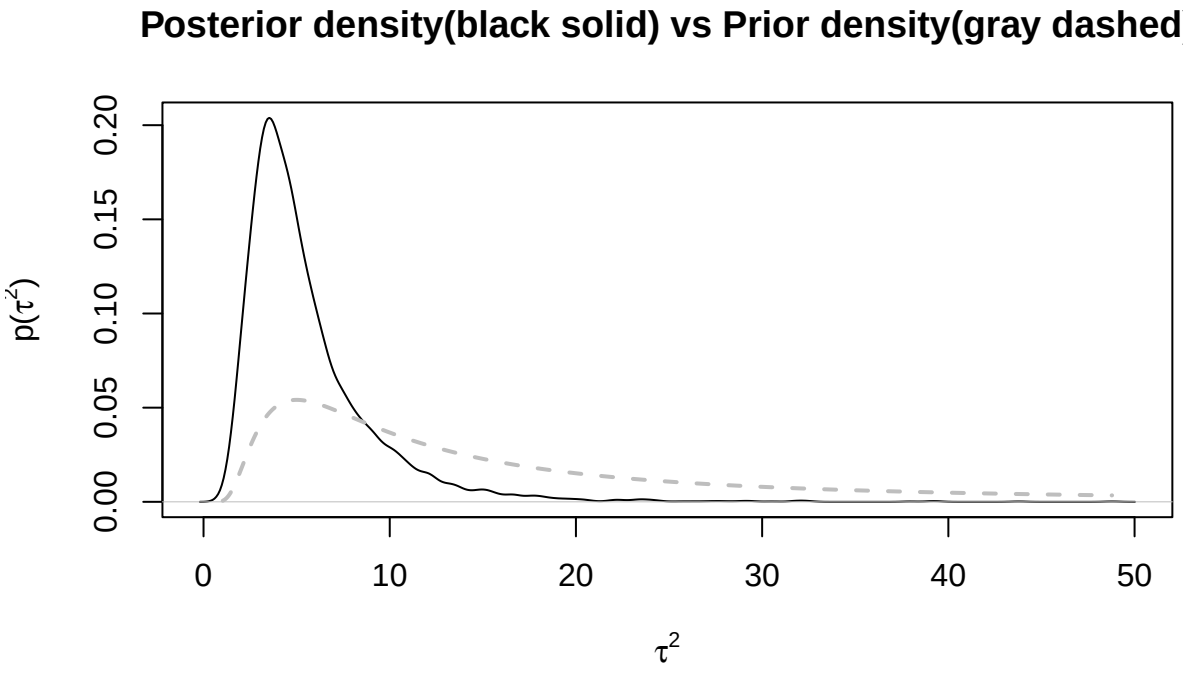
```
plot(density(MST[B:S,1]), main = "Posterior density(black solid) vs Prior density(gray dashed)",
     xlab = expression(mu), ylab = expression(paste("p(", mu, ")", sep="")))
x_mu <- seq(min(MST[B:S,1]), max(MST[B:S,1]), length.out = S)
lines(x_mu,dnorm(x_mu,mean=mu0,sd=sqrt(g20)),type="l",col="gray",lwd=2,lty=2)
```




```
dinvgamma<-function(x,a,b) {
ld<- a*log(b) -lgamma(a) -(a+1)*log(x)  -b/x
exp(ld)
}

plot(density(MST[B:S,2]), main = "Posterior density(black solid) vs Prior density(gray dashed)",
     xlab = expression(sigma^2), ylab = expression(paste("p(", sigma^2, ")", sep="")))
x_sigma <- seq(min(MST[B:S,2]), max(MST[B:S,2]), length.out = S)
lines(x_sigma,dinvgamma(x_sigma,a=nu0/2,b=(nu0*s20)/2),type="l",col="gray",lwd=2,lty=2)
```




```
plot(density(MST[B:S,3]), main = "Posterior density(black solid) vs Prior density(gray dashed)",
     xlab = expression(tau^2), ylab = expression(paste("p(", tau^2, ")", sep="")))
x_tau <- seq(min(MST[B:S,3]), max(MST[B:S,3]), length.out = S)
lines(x_tau,dinvgamma(x_tau,a=eta0/2,b=(eta0*t20)/2),type="l",col="gray",lwd=2,lty=2)
```

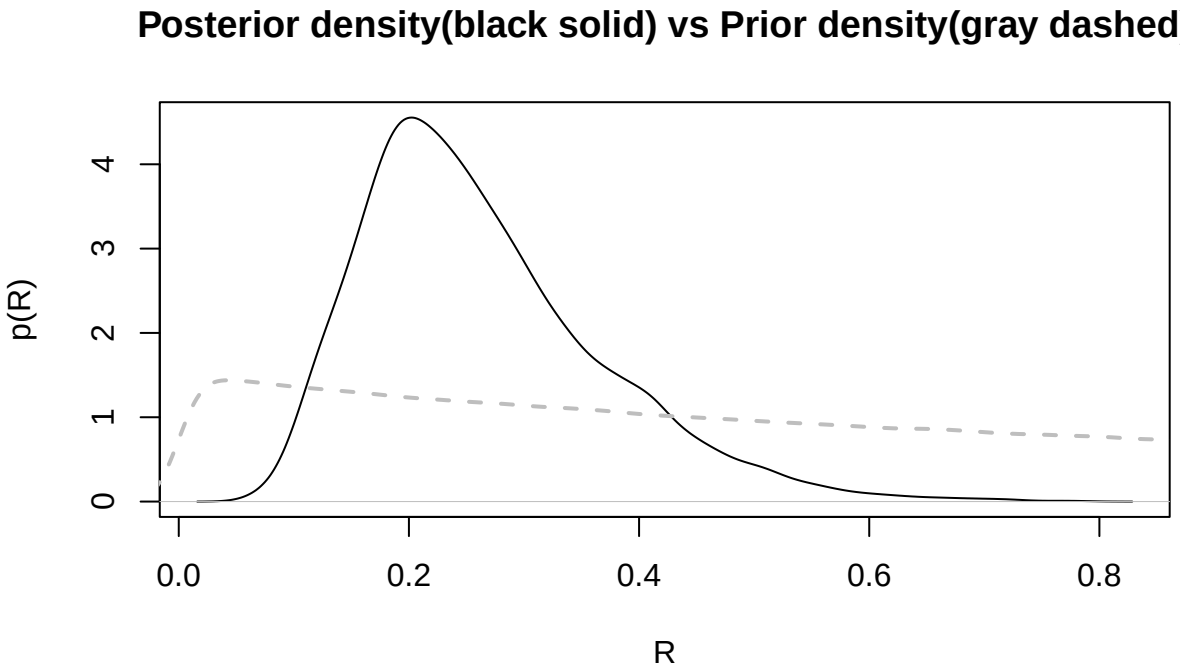


The posterior distributions are more concentrated, having less variance compared to prior.

Part c

c) Plot the posterior density of $R = \frac{\tau^2}{\sigma^2+\tau^2}$ and compare it to a plot of the prior density of R. Describe the evidence for between-school variation.

```
t20_sample = (1 / rgamma(1000000, eta0 / 2, eta0 * t20 / 2))
s20_sample = (1 / rgamma(1000000, nu0 / 2, nu0 * s20 / 2))

R0_sample =  (t20_sample) / (s20_sample+ t20_sample)

R_sample = MST[B:S, 3] / (MST[B:S, 2] + MST[B:S, 3])

plot(density(R_sample), main = "Posterior density(black solid) vs Prior density(gray dashed)",
     xlab = expression(R), ylab = expression(paste("p(R)")))
lines(density(R0_sample), col="Gray", lwd=2, lty=2)
```



If $\sigma^2 \gg \tau^2$, $R \to 0$, the within-school variance is the dominating factor for the variance in data; If $\tau^2 \gg \sigma^2$, $R \to 1$, the between-school variance is the dominating factor for the variance in data. R indicate the proportion of between-school variance in the total variance of data. We have a weak prior on which part is dominating before inference, but after calculating the posterior distribution of R, we know that around 20% of the variance in data comes from the between-school variance.

Part d

d) Obtain the posterior probability that θ_7 is smaller than θ_6, as well as the posterior probability that θ_7 is the smallest of all the θ's.

```
# The posterior probability that theta_7 is smaller than theta_6
theta7_lt_6 <- THETA[B:S, 7] < THETA[B:S, 6]
mean(theta7_lt_6)
```

```
## [1] 0.5085417
```

```
# The posterior probability that theta_7 is smaller than all
theta7_lt_all <- THETA[B:S, 7] <= THETA[B:S, -7]
theta7_lt_all <- apply(theta7_lt_all, MARGIN=1, FUN=all)
mean(theta7_lt_all)
```

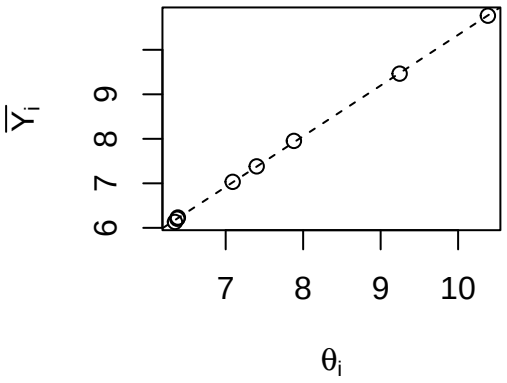
```
## [1] 0.3058333
```

$$\text{Posterior } P(\theta_7 < \theta_6) = 50.85\%$$
$$\text{Posterior } P(\theta_7 \leq \theta_.) = 30.58\%$$

Part e

e) Plot the sample averages $\overline{y}_1, \ldots, \overline{y}_8$ against the posterior expectations of $\theta_1, \ldots, \theta_8$, and describe the relationship. Also compute the sample mean of all observations and compare it to the posterior mean of μ.

```
thetabar <- apply(THETA[B:S,], MARGIN=2, FUN=mean)
plot(thetabar,ybar, xlab=expression(theta[i]),
ylab=expression(bar(Y[i])))
abline(lm(ybar~thetabar), lty=2)
```




```
# The sample mean of all observations
mean(Y[,2])
```

```
## [1] 7.691278
```

```
# Posterior mean of mu
mean(MST[B:S,1])
```

```
## [1] 7.551726
```

There is a linear relationship between θ_i and $\overline{y_i}$. The sample mean of all observations is 7.69, The posterior mean of μ is 7.55.

Problem 2: Exercise 9.3

Crime: The file crime.dat contains crime rates and data on 15 explanatory variables for 47 U.S. states, in which both the crime rates and the explanatory variables have been centered and scaled to have variance 1. A description of the variables can be obtained by typing library(MASS);?UScrime in R.

Part a

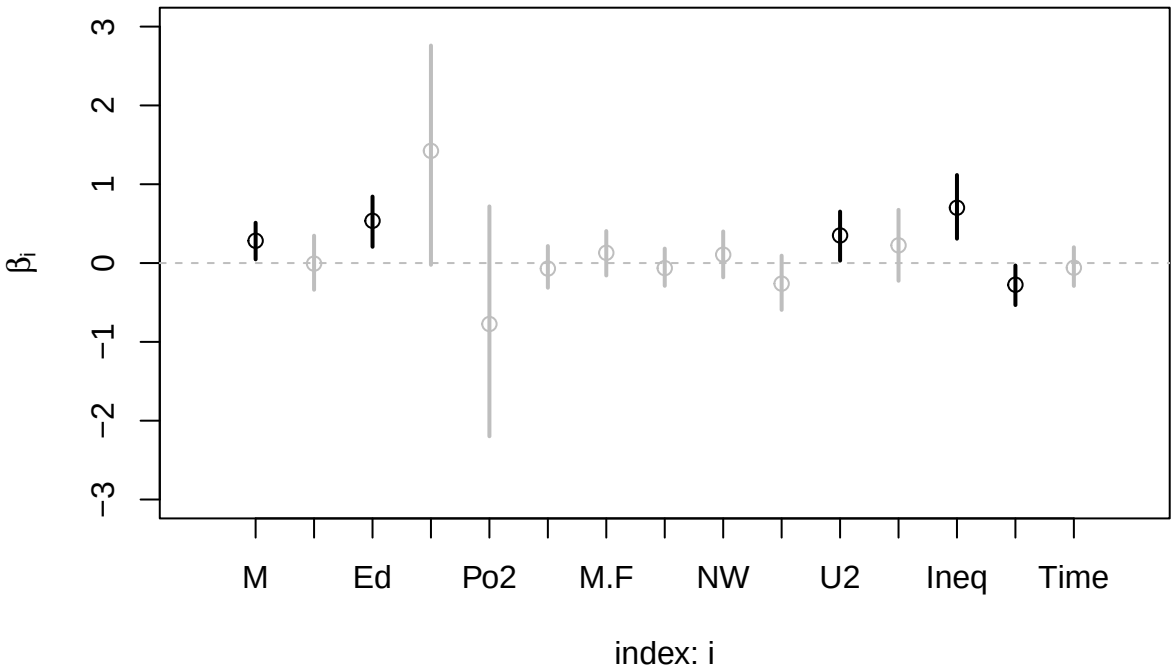
a) Fit a regression model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ using the g-prior with $g = n, \nu_0 = 2$ and $\sigma_0^2 = 1$. Obtain marginal posterior means and 95% confidence intervals for $\boldsymbol{\beta}$, and compare to the least squares estimates. Describe the relationships between crime and the explanatory variables. Which variables seem strongly predictive of crime rates?

From textbook page 158, under g-prior distributionn, $p(\sigma^2|\mathbf{y}, \mathbf{X})$ and $p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \sigma^2)$ are inverse-gamma and multivariate normal distributions respectively. A sample value of $\sigma^2, \boldsymbol{\beta}$ from the joint posterior distributions $p(\sigma^2, \boldsymbol{\beta}|\mathbf{y}, \mathbf{X})$ can be made with Monte Carlo approximation as follows:

1. sample $\frac{1}{\sigma^2} \sim \text{gamma}\left([\nu_0 + n]/2, [\nu_0\sigma_0^2 + \text{SSR}_g]/2\right)$; where $\text{SSR}_g = \mathbf{y}^T(\mathbf{I} - \frac{g}{g+1}\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)\mathbf{y}$
2. sample $\boldsymbol{\beta} \sim \text{multivariate normal}\left(\frac{g}{g+1}\hat{\boldsymbol{\beta}}_{\text{ols}}, \frac{g}{g+1}\sigma^2[\mathbf{X}^T\mathbf{X}]^{-1}\right)$

```
crime <- read.table("crime.dat", header=TRUE)
# crime.df <- as.data.frame(read.table("crime.dat", header=TRUE))

beta_bayes <- function(y, X){

n <- dim(X)[1] # rows of data
p <- dim(X)[2] # number of parameters

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))
return(beta)
}

y <- crime$y
X <- as.matrix(subset(crime, select = -c(y)))
beta <- beta_bayes(y,X)
```

```
CIbeta <- apply(beta, MARGIN=2, quantile, prob=c(0.025,0.5,0.975))
Significant <- apply(CIbeta, MARGIN=2,
    function(q){return(ifelse(q[1]<0 & q[3]>0, FALSE, TRUE))})
Meanbeta <- apply(beta, MARGIN=2, mean)
LSbeta <- lm(y~X)$coeff[-1]
CIbeta <- rbind(CIbeta, Significant, Meanbeta, LSbeta)
# xtable(CIbeta)
p <- dim(X)[2] # number of parameters
plot(1:p,type="n", ylab=expression(beta[i]), xlab="index: i",
     xlim=c(0,p+1), ylim=c(-3,3), xaxt="n")
for (i in 1:p){
color=ifelse(CIbeta[4,i],"black","gray")
segments(i,CIbeta[1,i],i,CIbeta[3,i], lwd=2,  col=color)
#points(i, CIbeta[1, i], pch=4, col=color)
points(i, CIbeta[2, i], col=color)
#points(i, CIbeta[3, i], pch=4, col=color)
}
axis(side = 1, at = seq(1,p,by=1), labels = colnames(CIbeta))
abline(h=0, lty=2, col="GRAY")
```



	M	So	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	GDP	Ineq	Prob	Time
2.5%	0.05	-0.34	0.21	-0.02	-2.20	-0.31	-0.16	-0.29	-0 .18	-0.59	0.03	-0.22	0.31	-0.53	-0.29
50%	0.28	-0.01	0.54	1.42	-0.77	-0.07	0.13	-0.06	0.1 1	-0.26	0.35	0.22	0.70	-0.27	-0.06
97.5%	0.51	0.35	0.84	2.76	0.72	0.22	0.41	0.18	0.40	0.09	0.65	0.68	1.12	-0.03	0.20
Significant	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	1.00	1.00	0.00
Meanbeta	0.28	0.00	0.53	1.42	-0.74	-0.06	0.13	-0.07	0.11	-0.26	0.35	0.23	0.70	-0.28	-0.06
LSbeta	0.29	-0.00	0.54	1.47	-0.78	-0.07	0.13	-0.07	0 .11	-0.27	0.37	0.24	0.73	-0.29	-0.06

In the table above, 2.5% and 97.5% rows gives the lower and upper bound of 95% credible intervals, Significant = 1 indicates variables are significant (does not include 0 in the credible intervals). The significant variables selected are: M, ED, U2, Ineq, Prob. "Meanbeta" row gives the marginal posterior means. The last row "LSbeta" gives the classic Least Square Regression result.

Part b

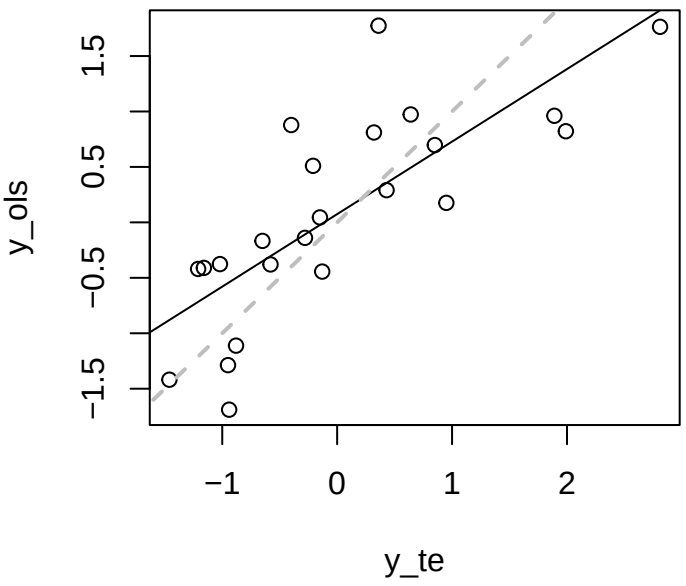
b) Lets see how well regression models can predict crime rates based on the $\mathbf{X}$-variables. Randomly divide the crime roughly in half, into a training set $\{\mathbf{y}_{\text{tr}}, \mathbf{X}_{\text{tr}}\}$ and a test set $\{\mathbf{y}_{\text{te}}, \mathbf{X}_{\text{te}}\}$

```
set.seed(1)
train_i = sample.int(length(y), size = round(length(y) / 2), replace = FALSE)
ytr = y[train_i]
Xtr = X[train_i, ]
yte = y[-train_i]
Xte = X[-train_i, ]
```

i. Using only the training set, obtain least squares regression coefficients $\hat{\boldsymbol{\beta}}_{\text{ols}}$. Obtain predicted values for the test data by computing $\hat{\mathbf{y}}_{\text{ols}}$. Plot $\hat{\mathbf{y}}_{\text{ols}}$ versus $\mathbf{y}_{\text{te}}$ and compute the prediction error $\frac{1}{n_{\text{te}}} \sum (y_{i,\text{te}} - \hat{y}_{i,\text{ols}})^2$

The dashed grey line is the reference line for $y_{te} = y_{ols}$

```
Beta_ols = solve(t(Xtr) %*% Xtr) %*% t(Xtr) %*% ytr
#Output Beta_ols
#xtable(t(Beta_ols))

y_ols = Xte %*% Beta_ols
xy_range <- range(c(-1.5, 3))
plot(yte, y_ols, xlab="y_te", ylab="y_ols")
abline(lm(y_ols~yte))
# line y=x
ref_range<-seq(-1.6,3,by=0.1)
lines(ref_range,ref_range,
      type="l", lty=2, lwd=2, col="GRAY")
```



```
#abline(a=0,b=1,lty=2,lwd=2,col="GRAY")
```

$\hat{\beta}_{\text{ols}}$ is:

	M	So	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	GDP	In eq	Prob	Time
1	0.19	0.21	0.64	0.31	0.45	-0.09	0.04	0.08	-0.01	-0.03	0.15	0.10	0.77	-0.26	0.06

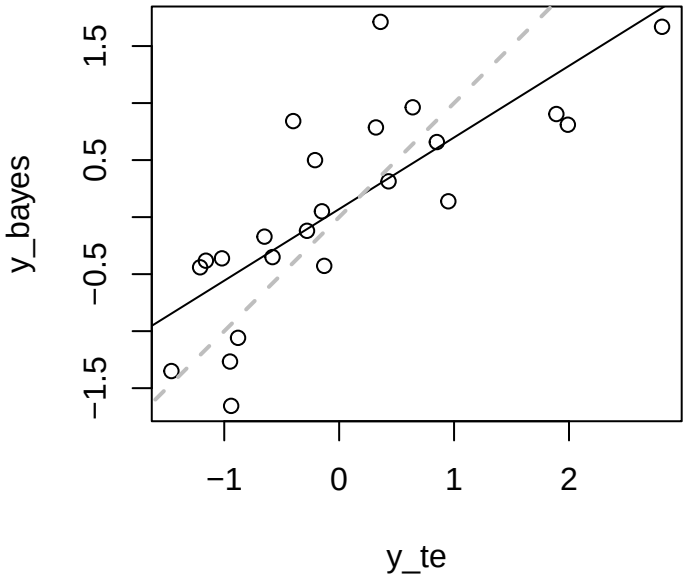
Prediction Error:

```
pred_error = sum((yte - y_ols)^2) / length(yte)
pred_error
```

```
## [1] 0.4880959
```

ii. Now obtain the posterior mean $\hat{\beta}_{\text{Bayes}} = E[\beta \mid \mathbf{y}_{\text{tr}}]$ using the g-prior described above and the training data only. Obtain predictions for the test set $\hat{\mathbf{y}}_{\text{Bayes}} = \mathbf{X}_{\text{test}}\hat{\beta}_{\text{Bayes}}$. Plot versus the test data, compute the prediction error, and compare to the OLS prediction error. Explain the results.

```
Beta_bayes = as.matrix(colMeans(beta_bayes(ytr,Xtr)))
#Output Beta_bayes
#xtable(t(Beta_bayes))

y_bayes = Xte %*% Beta_bayes

plot(yte, y_bayes, xlab="y_te", ylab="y_bayes")
abline(lm(y_bayes~yte))
# line y=x
ref_range<-seq(-1.6,3,by=0.1)
lines(ref_range,ref_range,
type="l", lty=2, lwd=2, col="GRAY")
```



$\hat{\beta}_{\text{Bayes}}$ is:

	M	So	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	GDP	In eq	Prob	Time
1	0.18	0.22	0.61	0.23	0.48	-0.09	0.04	0.09	-0.02	-0.03	0.14	0.11	0.74	-0.26	0.06

```
pred_error = sum((yte - y_bayes)^2) / length(yte)
pred_error
```

```
## [1] 0.4910634
```

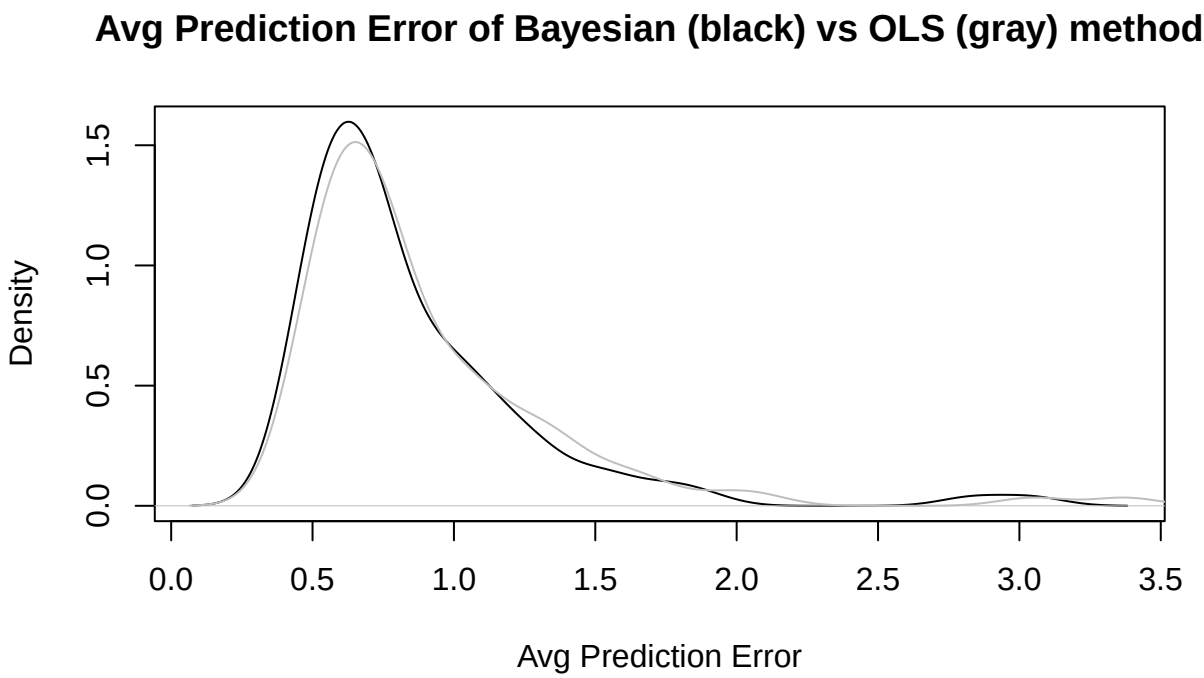
Runing one single test, we cannot see significant difference between average prediction error of OLS (0.488) vs. Bayesian methods (0.491).

Part c

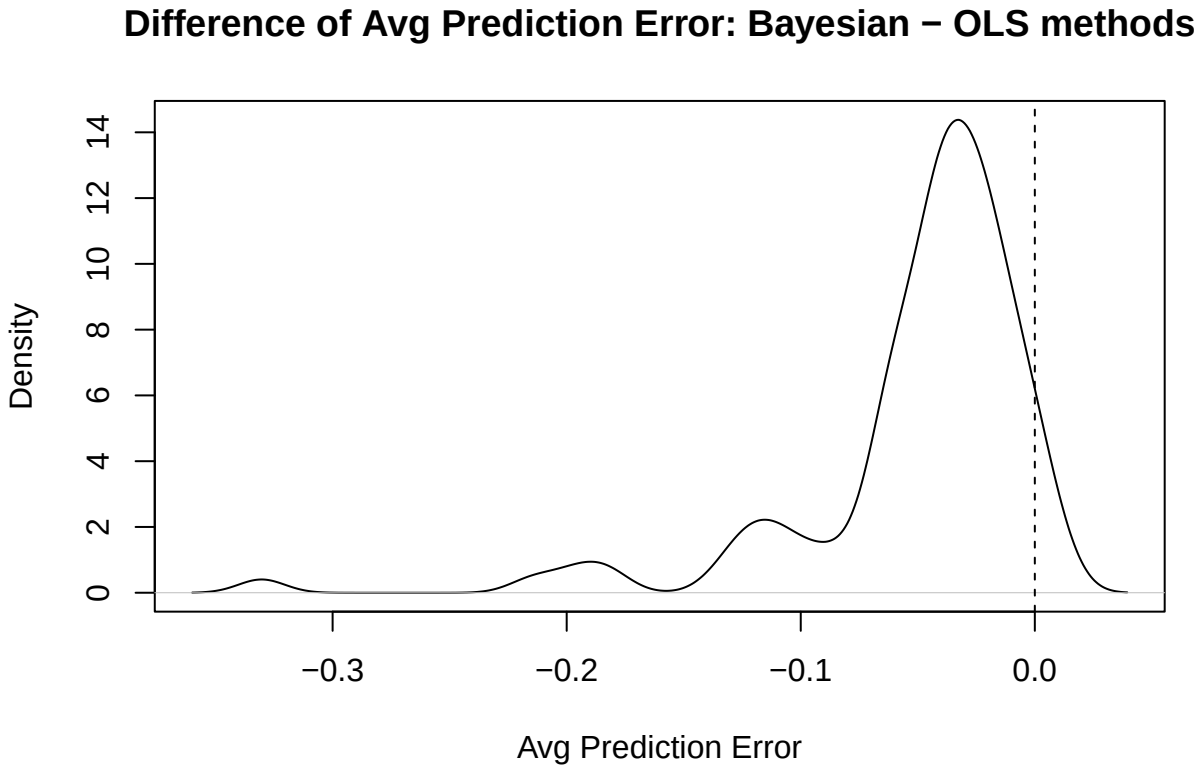
c) Repeat the procedures in b) many times with different randomly generated test and training sets. Compute the average prediction error for both the OLS and Bayesian methods.

```
N = 100
set.seed(1)
pred_errors = t(sapply(1:N, function(i) {
  y = crime$y
  X <- as.matrix(subset(crime, select = -c(y)))
  train_i = sample.int(length(y), size = round(length(y) / 2), replace = FALSE)
  ytr = y[train_i]
  Xtr = X[train_i, ]
  yte = y[-train_i]
  Xte = X[-train_i, ]

  # OLS
  Beta_ols = solve(t(Xtr) %*% Xtr) %*% t(Xtr) %*% ytr
  y_ols = Xte %*% Beta_ols
  pred_error_ols = sum((yte - y_ols)^2) / length(yte)

  # Bayes
  y = ytr
  X = Xtr
  Beta_bayes = as.matrix(colMeans(beta_bayes(ytr,Xtr)))
  y_bayes = Xte %*% Beta_bayes
  pred_error_bayes = sum((yte - y_bayes)^2) / length(yte)

  c(pred_error_ols, pred_error_bayes)
}))
colnames(pred_errors) = c('ols', 'bayes')
```

```
plot(density(pred_errors[,2]), xlab="Avg Prediction Error",
     main="Avg Prediction Error of Bayesian (black) vs OLS (gray) methods")
lines(density(pred_errors[,1]), col="gray")
```




```
plot(density(pred_errors[,2]-pred_errors[,1]), xlab="Avg Prediction Error",
     main="Difference of Avg Prediction Error: Bayesian - OLS methods")
abline(v=0, lty=2)
```



Bayesian method are more likely to have less prediction error then OLS method, especially when the prediction errors are large.

Problem 3: Exercise 10.5

Logistic regression variable selection: Consider a logistic regression model for predicting diabetes as a function of x_1 = number of pregnancies, x_2 = blood pressure, x_3 = body mass index, x_4 = diabetes pedigree and x_5 = age. Using the data in azdiabetes.dat, center and scale each of the x-variables by subtracting the sample average and dividing by the sample standard deviation for each variable. Consider a logistic regression model of the form $\Pr(Y_i = 1|\mathbf{x}_i, \boldsymbol{\beta}, \boldsymbol{\gamma}) = e^{\theta_i}/(1 + e^{\theta_i})$ where

$$\theta_i = \beta_0 + \beta_1\gamma_1 x_{i,1} + \beta_2\gamma_2 x_{i,2} + \beta_3\gamma_3 x_{i,3} + \beta_4\gamma_4 x_{i,4} + \beta_5\gamma_5 x_{i,5}.$$

In this model, each γ_j is either 0 or 1, indicating whether or not variable j is a predictor of diabetes. For example, if it were the case that $\gamma = (1, 1, 0, 0, 0)$, then $\theta_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2}$. Obtain posterior distributions for β and γ, using independent prior distributions for the parameters, such that $\gamma_j \sim \text{binary}(\frac{1}{2})$, $\beta_0 \sim \text{normal}(0, 16)$ and $\beta_j \sim \text{normal}(0, 4)$ for each $j > 0$.

Part a

a) Implement a Metropolis-Hastings algorithm for approximating the posterior distribution of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. Examine the sequences $\beta_j^{(s)}$ and $\beta_j^{(s)} \times \gamma_j^{(s)}$ for each j and discuss the mixing of the chain.

We iterativly go through step 1 to step 11 below to sample $\beta_0, \ldots, \beta_5, \gamma_1, \ldots, \gamma_5$ using Metropolis-Hastings algorithms:

1. Update β_0:
   a. Sample $\beta_0^* \sim N(\beta_0^{(s)}, 16)$
   b. compute the log acceptance ratio[1][2]

$$log(r) = log\frac{p(\mathbf{Y}|\mathbf{X}, \beta_1^{(s)}, \ldots, \beta_5^{(s)}, \gamma_1^{(s)}, \ldots, \gamma_5^{(s)}, \beta_0^*)}{p(\mathbf{Y}|\mathbf{X}, \beta_1^{(s)}, \ldots, \beta_5^{(s)}, \gamma_1^{(s)}, \ldots, \gamma_5^{(s)}, \beta_0^{(s)})}.$$

   c. set $\beta_0^{(s+1)}$ to β_0^* or $\beta_0^{(s)}$ with probability $\min(1, r)$ and $\max(0, 1 - r)$.
2. Update β_1:
   a. Sample $\beta_1^* \sim N(\beta_1^{(s)}, 4)$
   b. compute the log acceptance ratio

$$log(r) = log\frac{p(\mathbf{Y}|\mathbf{X}, \beta_0^{(s+1)}, \beta_2^{(s)}, \ldots, \beta_5^{(s)}, \gamma_1^{(s)}, \ldots, \gamma_5^{(s)}, \beta_1^*)}{p(\mathbf{Y}|\mathbf{X}, \beta_0^{(s+1)}, \beta_2^{(s)}, \ldots, \beta_5^{(s)}, \gamma_1^{(s)}, \ldots, \gamma_5^{(s)}, \beta_1^{(s)})}.$$

   c. set $\beta_1^{(s+1)}$ to β_1^* or $\beta_1^{(s)}$ with probability $\min(1, r)$ and $\max(0, 1 - r)$.

...

11. Update γ_5:

   a. Sample $\gamma_5^* \sim B(n = 1, p = 0.5)$
   b. compute the log acceptance ratio

$$log(r) = log\frac{p(\mathbf{Y}|\mathbf{X}, \beta_0^{(s+1)}, \ldots, \beta_5^{(s+1)}, \gamma_1^{(s+1)}, \ldots, \gamma_4^{(s+1)}, \gamma_5^*)}{p(\mathbf{Y}|\mathbf{X}, \beta_0^{(s+1)}, \ldots, \beta_5^{(s+1)}, \gamma_1^{(s+1)}, \ldots, \gamma_4^{(s+1)}, \gamma_5^{(s)})}.$$

   c. set $\gamma_5^{(s+1)}$ to γ_5^* or $\gamma_5^{(s)}$ with probability $\min(1, r)$ and $\max(0, 1 - r)$.

[1] In many cases, computing the ratio r directly can be numerically unstable, a problem that often can be remedied by computing the logarithm of r

[2] We are using symmeric proposal distribution, so the proposal distribution canceled out

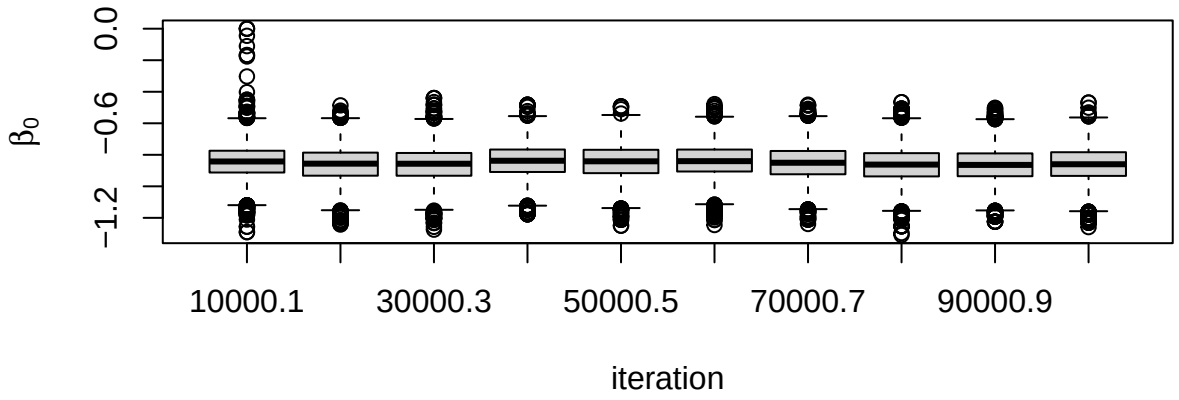
```
azdiabetes <- read.table("azdiabetes.dat", header=TRUE)
X <- scale(azdiabetes[, c("npreg", "bp", "bmi", "ped", "age")])
n <- dim(X)[1]
X <- cbind(rep(1,n),X)
Y <- azdiabetes$diabetes == "Yes"

sigmoid <- function(x){
    return(1/(1+exp(-x)))
}
GAMMA<-BETA<-NULL
delta=2
BETA <- rbind(BETA, rep(0,6))
GAMMA <- rbind(GAMMA, rep(0,5))
delta <- c(4,2,2,2,2,2)
S=100000
ACCEPT.BETA.count <- rep(0,6)
ACCEPT.GAMMA.count <- rep(0,5)
set.seed(8964)
for(i in 1:S){
Beta_update<-BETA[i,]
Gamma_update<-GAMMA[i,]

# updating beta
for(j in 1:6){
  beta.p <- Beta_update
  beta.p[j] <- rnorm(1, Beta_update[j], delta[j])
    lr<-sum(dbinom(Y,size=1,
           prob=sigmoid(X%*%(beta.p*c(1,Gamma_update))),log=TRUE ) -
        dbinom(Y,size=1,
           prob=sigmoid(X%*%(Beta_update*c(1,Gamma_update))),log=TRUE ) )
    if( log(runif(1))<lr ) {
        ACCEPT.BETA.count[j] <- ACCEPT.BETA.count[j]+1
        Beta_update<-beta.p }
}# End update beta

# update Gamma
for(j in 1:5){
  gamma.p <- Gamma_update
  gamma.p[j] <- sample(0:1, size = 1)
  lr<-sum(dbinom(Y,size=1,
          prob=sigmoid(X%*%(Beta_update*c(1,gamma.p))),log=TRUE ) -
      dbinom(Y,size=1,
          prob=sigmoid(X%*%(Beta_update*c(1,Gamma_update))),log=TRUE ) )
    if( log(runif(1))<lr ) {
        ACCEPT.GAMMA.count[j] <- ACCEPT.GAMMA.count[j]+1
        Gamma_update<-gamma.p}
}# End update beta

BETA <- rbind(BETA, Beta_update)
GAMMA <- rbind(GAMMA, Gamma_update)
print(i)
print(Beta_update)
```

```
print(Gamma_update)
}
```

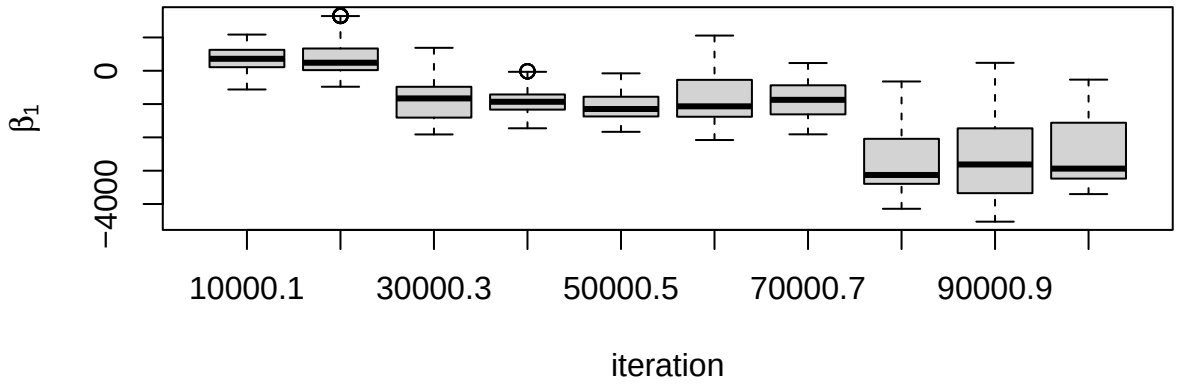
```
save(BETA, file="BETA.10.5.RData")
save(GAMMA, file="GAMMA.10.5.RData")
save(ACCEPT.BETA.count, file="ACCEPT.BETA.10.5.count.RData")
save(ACCEPT.GAMMA.count, file="ACCEPT.GAMMA.10.5.count.RData")
```

```
load("BETA.10.5.RData")
load("GAMMA.10.5.RData")
load("ACCEPT.BETA.10.5.count.RData")
load("ACCEPT.GAMMA.10.5.count.RData")
```

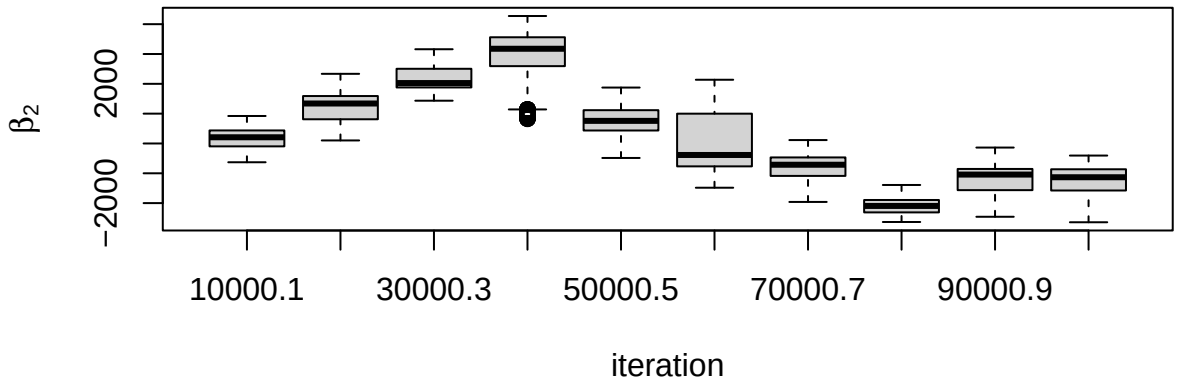
```
stationarity.plot(BETA[,1],xlab="iteration",ylab=expression(beta[0]))
```



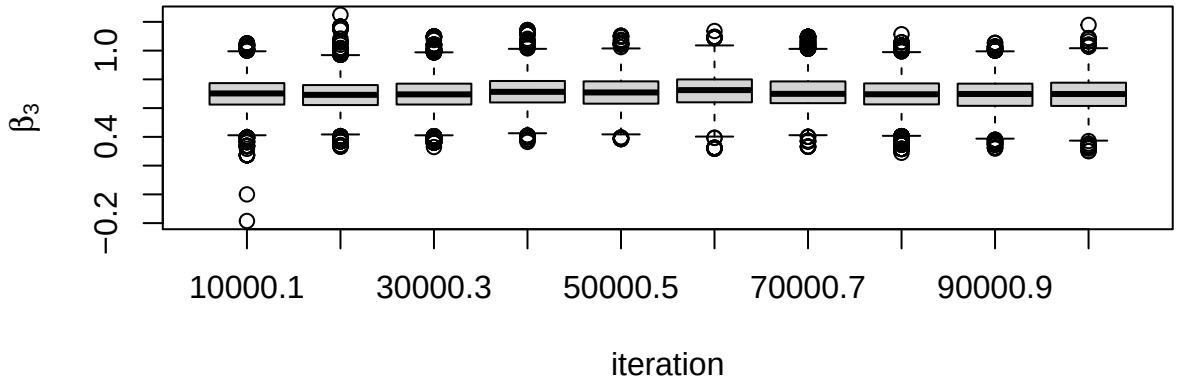

```
stationarity.plot(BETA[,2],xlab="iteration",ylab=expression(beta[1]))
```



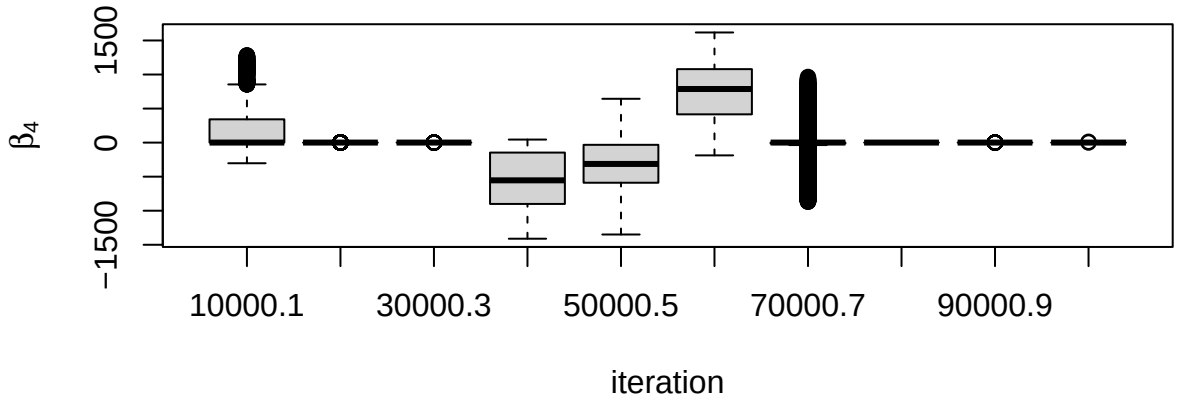

```
stationarity.plot(BETA[,3],xlab="iteration",ylab=expression(beta[2]))
```



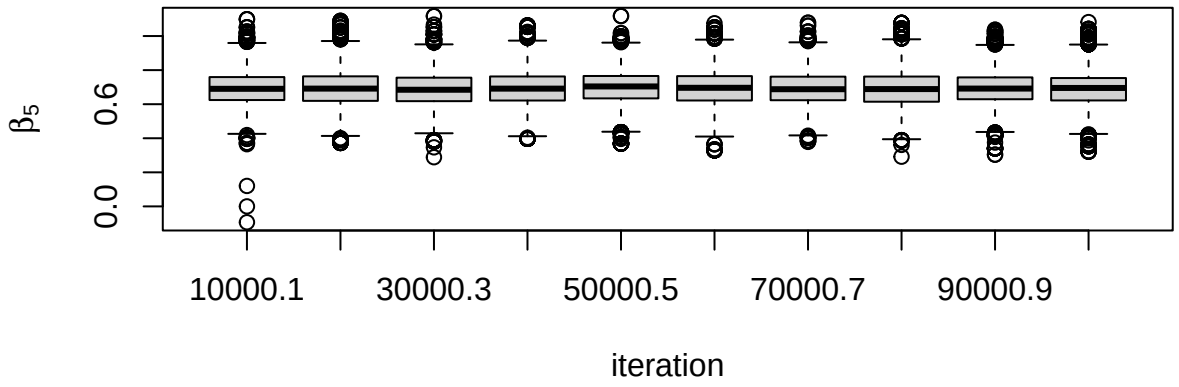

```
stationarity.plot(BETA[,4],xlab="iteration",ylab=expression(beta[3]))
```



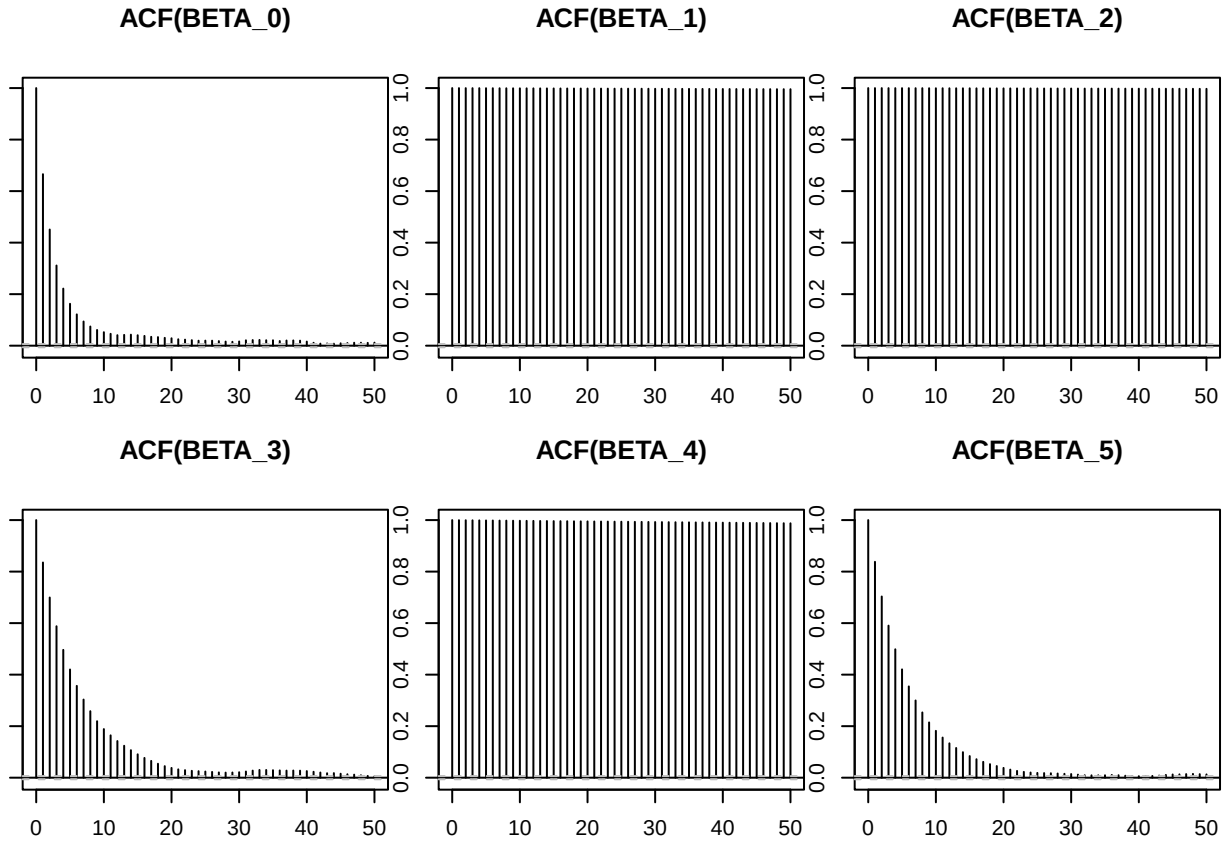

```
stationarity.plot(BETA[,5],xlab="iteration",ylab=expression(beta[4]))
```




```
stationarity.plot(BETA[,6],xlab="iteration",ylab=expression(beta[5]))
```




```
par(mfrow = c(2, 3), mar = c(3, 1, 3, 1))
acf(BETA[,1],ci.col="gray",xlab="lag", main = 'ACF(BETA_0)')
acf(BETA[,2],ci.col="gray",xlab="lag", main = 'ACF(BETA_1)')
acf(BETA[,3],ci.col="gray",xlab="lag", main = 'ACF(BETA_2)')
acf(BETA[,4],ci.col="gray",xlab="lag", main = 'ACF(BETA_3)')
acf(BETA[,5],ci.col="gray",xlab="lag", main = 'ACF(BETA_4)')
acf(BETA[,6],ci.col="gray",xlab="lag", main = 'ACF(BETA_5)')
```




```
ACCEPT.BETA.count
```

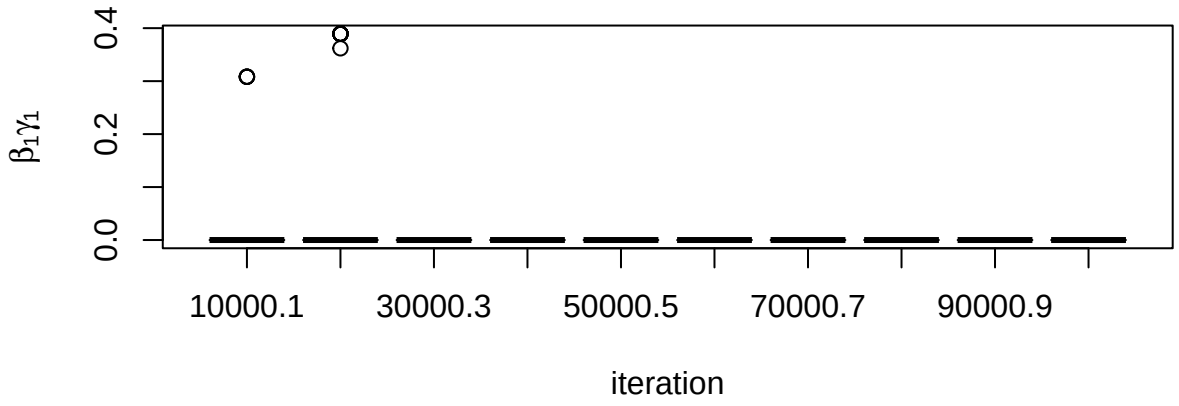
```
## [1] 43826 99985 99994 13603 36415 12782
```

```
ACCEPT.GAMMA.count
```

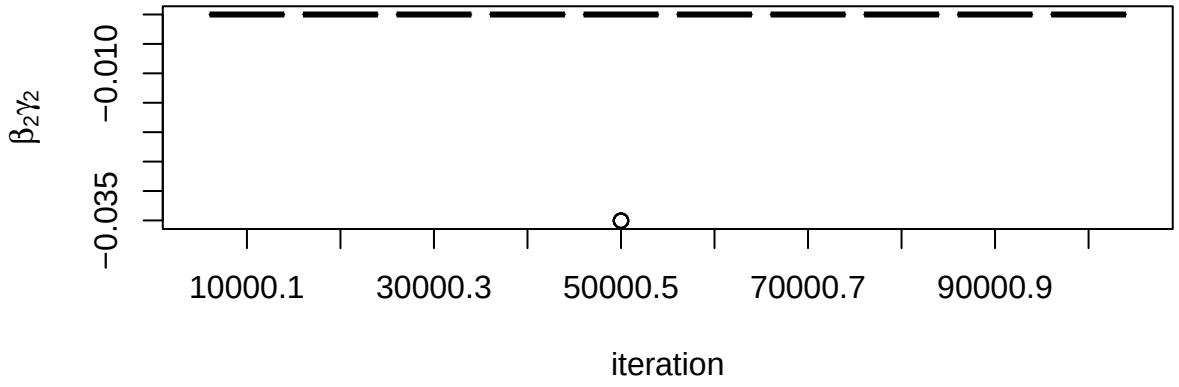
```
## [1] 49814 50207 50266 49971 50021
```

The following graphs are stationarity plots for $\beta_j \times \gamma_j$

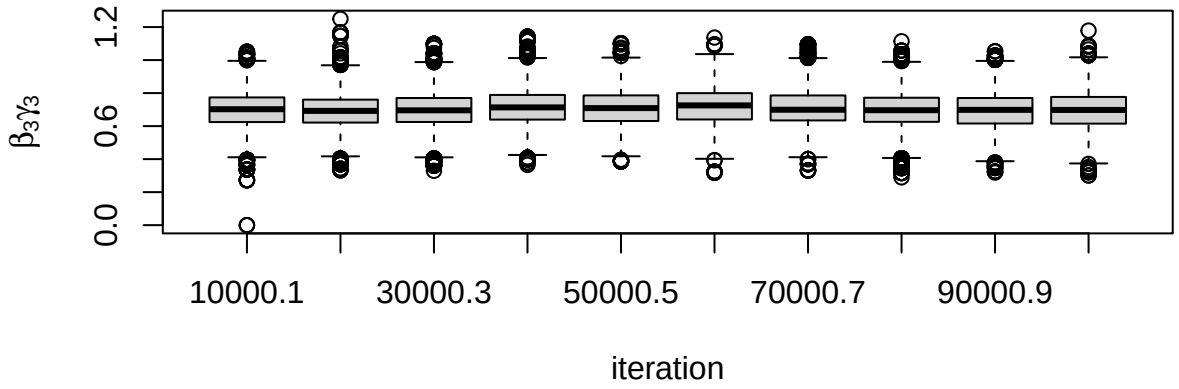
```
stationarity.plot(BETA[,2]*GAMMA[,1],xlab="iteration",ylab=expression(beta[1]*gamma[1]))
```



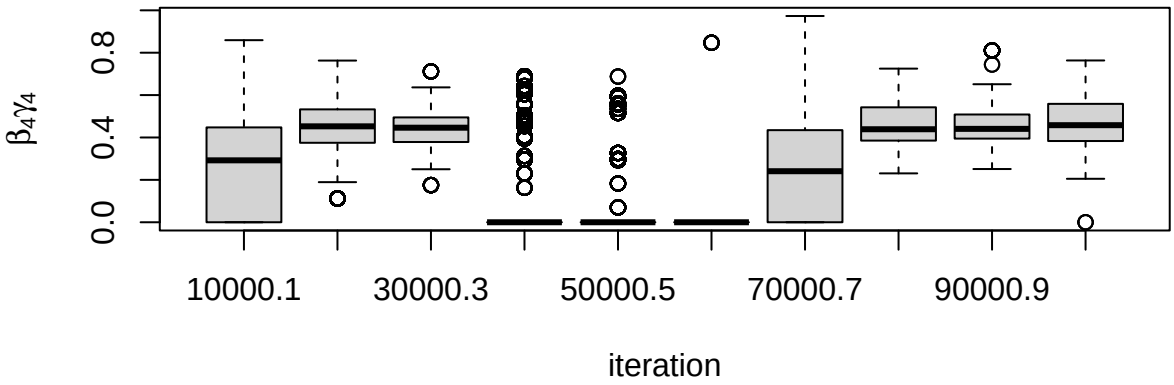

```
stationarity.plot(BETA[,3]*GAMMA[,2],xlab="iteration",ylab=expression(beta[2]*gamma[2]))
```



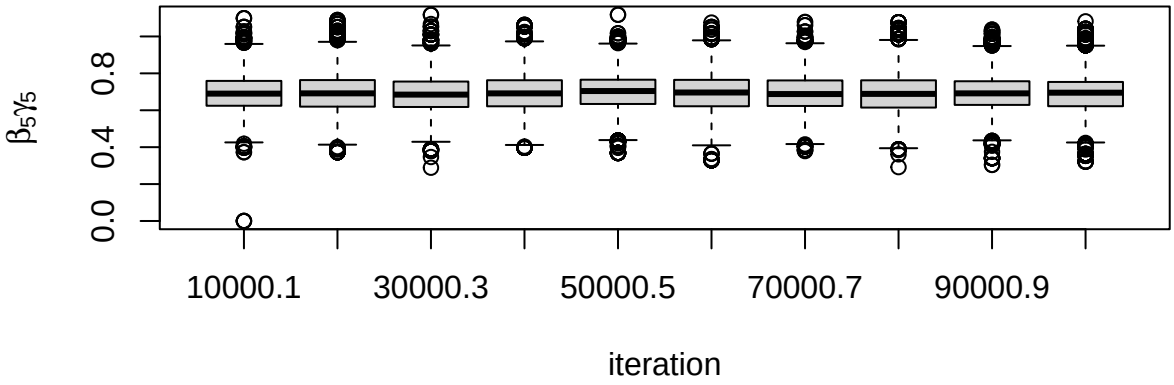

```
stationarity.plot(BETA[,4]*GAMMA[,3],xlab="iteration",ylab=expression(beta[3]*gamma[3]))
```




```
stationarity.plot(BETA[,5]*GAMMA[,4],xlab="iteration",ylab=expression(beta[4]*gamma[4]))
```




```
stationarity.plot(BETA[,6]*GAMMA[,5],xlab="iteration",ylab=expression(beta[5]*gamma[5]))
```



The sampling results are not very satisfying, only β_0 seems fine to me. β_1, β_2 shows too high of accptance rate and also too high of autocorrelation. But given that $\beta_1\gamma_1$ and $\beta_2\gamma_2$ are stationarity at 0, we can accept that. However, for $\beta_4\gamma_4$, it's not acheiving stationarity.

Part b

b) Approximate the posterior probability of the top five most frequently occurring values of γ. How good do you think the MCMC estimates of these posterior probabilities are?

```
# Convert each row to a character string
gamma_strings <- apply(GAMMA, 1, paste, collapse = "")
# Count occurrences of each vector
vector_counts <- table(gamma_strings)
# Sort the counts in decreasing order and select the top 5 most frequent vectors
sort(vector_counts, decreasing = TRUE)[1:5]
```

```
## gamma_strings
## 00111 00101 10111 01101 00000
## 64095 35881    15     6     3
```

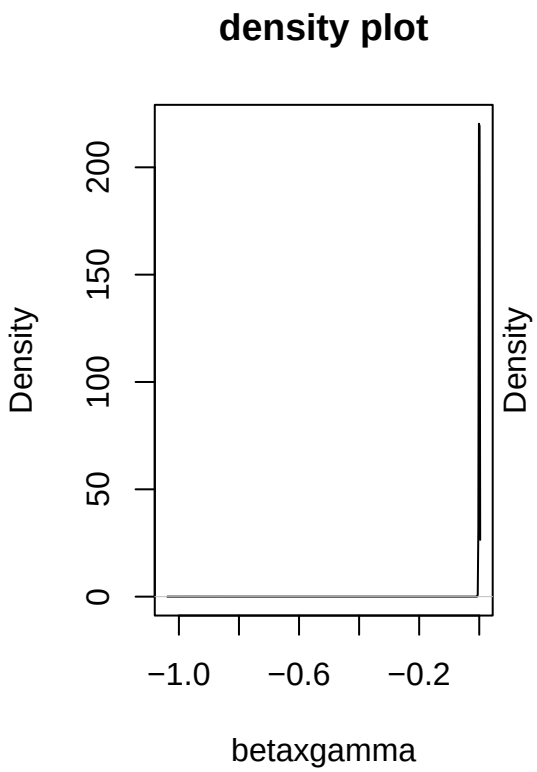
Part c

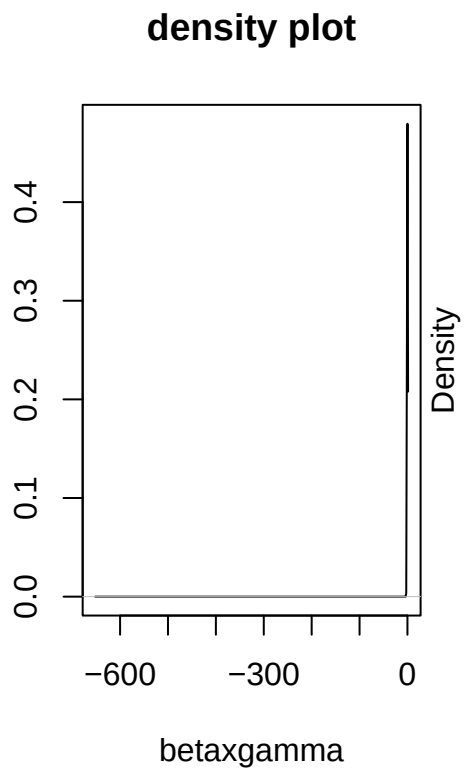
c) For each j, plot posterior densities and obtain posterior means for $\beta_j\gamma_j$. Also obtain $\Pr(\gamma_j = 1|\mathbf{x}, \mathbf{y})$.

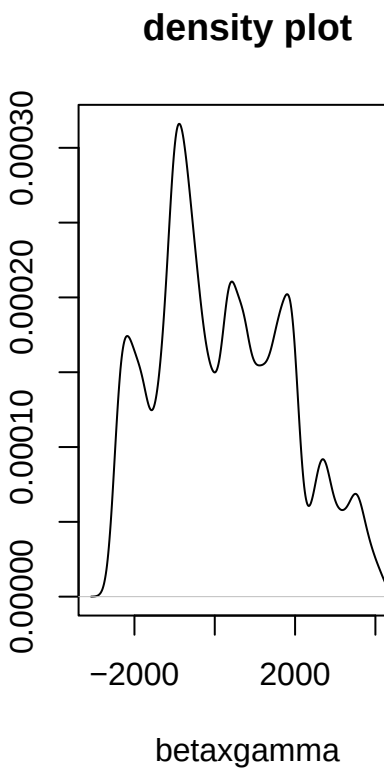
```r
# Burn-in first 3000
B<-1000
dim(GAMMA)
```

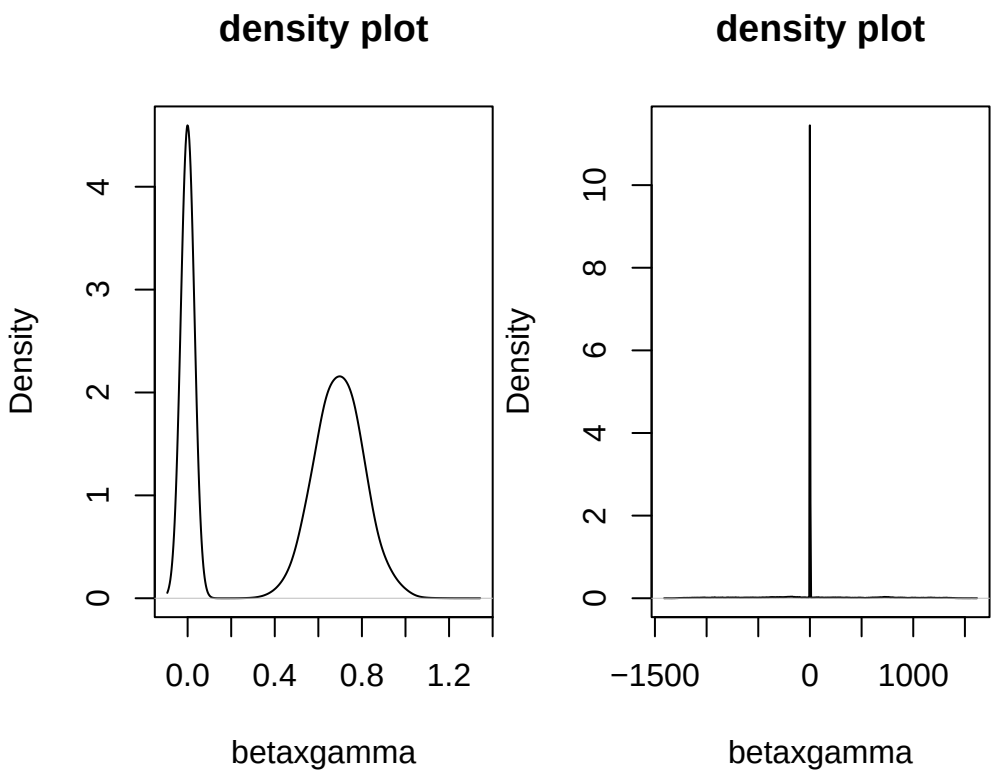
```
## [1] 100001      5
```

```r
# thinning every 30 samples
Gsave <- GAMMA[seq(B,dim(GAMMA)[1],by=30)]
Bsave <- BETA[seq(B,dim(BETA)[1],by=30)]
for (j in 1:5){
plot(density(GAMMA[,j]*BETA[,j]),xlab="betaxgamma", main="density plot")
}
```









Problem 4: Exercise 11.4

Hierarchical logistic regression: The Washington Assessment of Student Learning (WASL) is a standardized test given to students in the state of Washington. Letting j index the counties within the state of Washington and i index schools within counties, the file mathstandard.dat includes data on the following variables:
$y_{i,j}$ = the indicator that more than half the 10th graders in school i, j passed the WASL math exam;
$x_{i,j}$ = the percentage of teachers in school i, j who have a masters degree.
In this exercise we will construct an algorithm to approximate the posterior distribution of the parameters in a generalized linear mixed-effects model for these data. The model is a mixed effects version of logistic regression:

$$y_{i,j} \sim \text{binomial}(\frac{e^{\gamma_{i,j}}}{1+e^{\gamma_{i,j}}}), \text{ where } \gamma_{i,j} = \beta_{0,j} + \beta_{1,j}x_{i,j}$$
$$\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_J \sim \text{ i.i.d. multivariate normal } (\boldsymbol{\theta}, \Sigma), \text{ where } \boldsymbol{\beta}_j = (\beta_{0,j}, \beta_{1,j}).$$

Part a

a) The unknown parameters in the model include population-level parameters $\{\boldsymbol{\theta}, \Sigma\}$ and the group-level parameters $\{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_J\}$. Draw a diagram that describes the relationships between these parameters, the data $\{y_{i,j}, x_{i,j}, i = 1, \ldots, n_j, j = 1, \ldots, J\}$, and prior distributions.

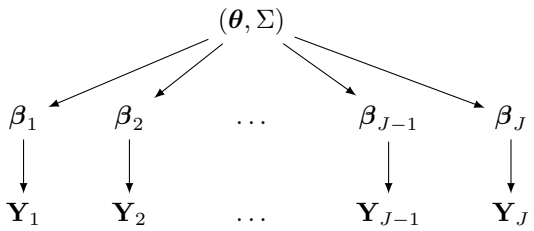


Figure 2: Graphical representation of the hierarchical model.

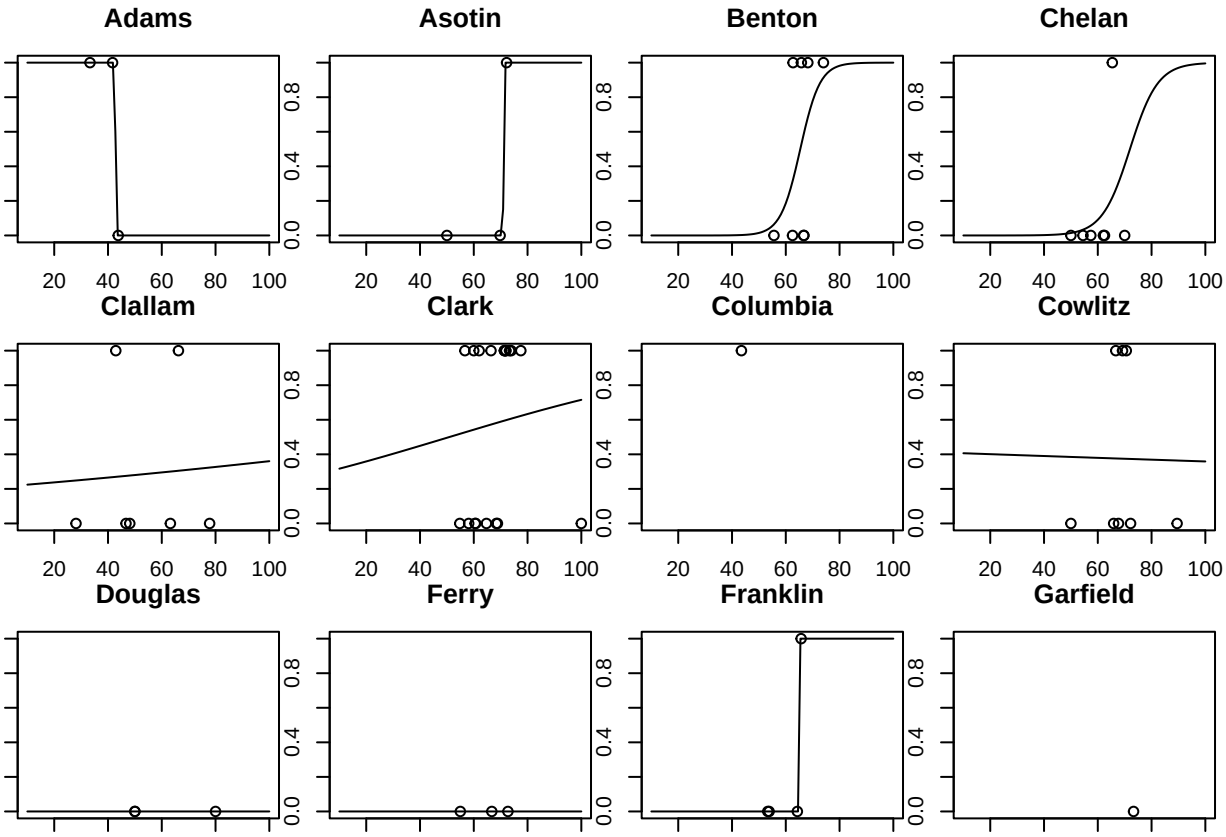
Part b

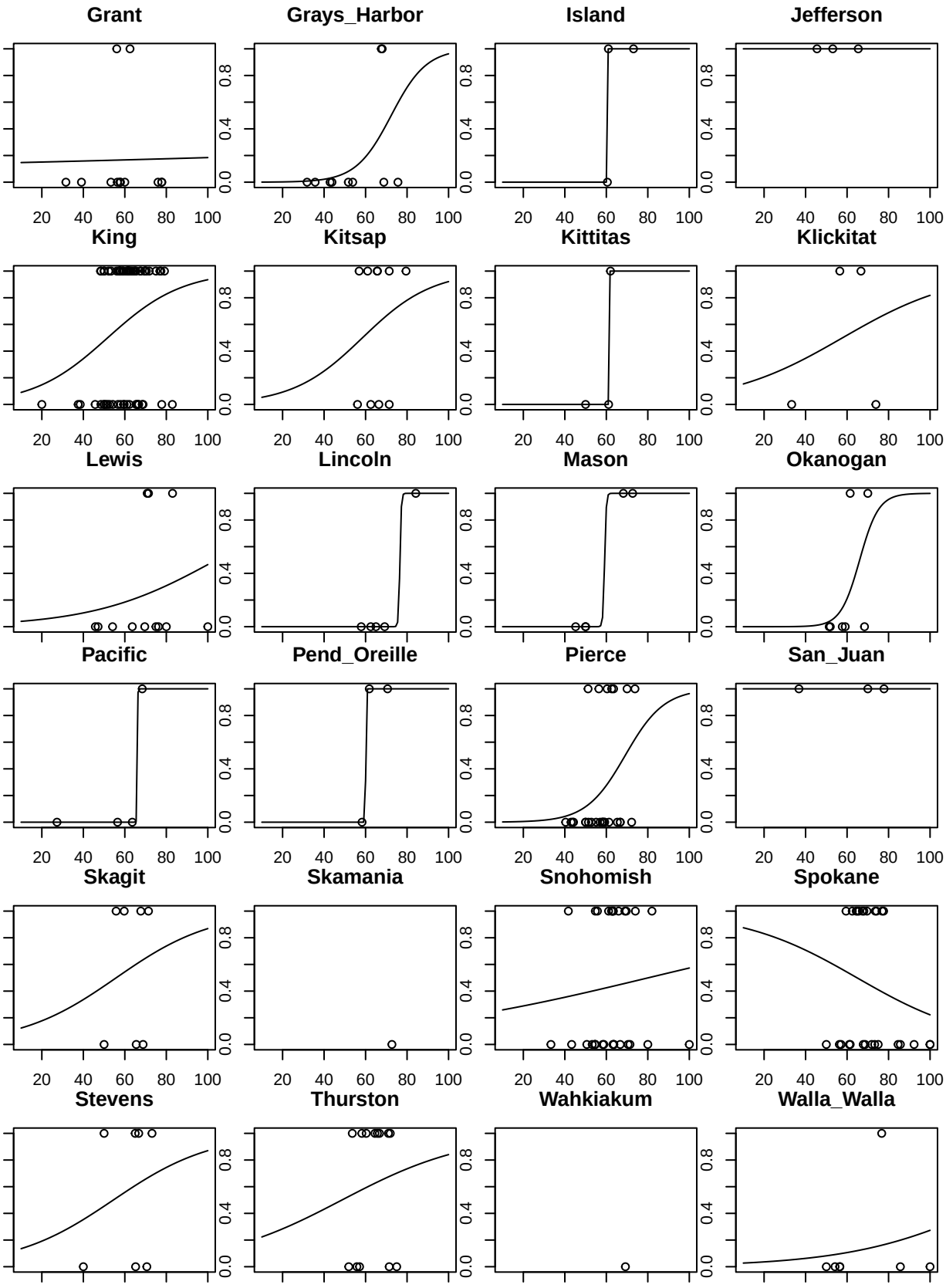
b) Before we do a Bayesian analysis, we will get some ad hoc estimates of these parameters via maximum likelihood: Fit a separate logistic regression model for each group, possibly using the **glm** command in R via beta.j <- glm(y.j $\tilde{\text{X}}$.j, family=binomial)$coef. Explain any problems you have with obtaining estimates for each county. Plot $\exp\{\hat{\beta}_{0,j} + \hat{\beta}_{1,j}x\}/(1 + \exp\{\hat{\beta}_{0,j} + \hat{\beta}_{1,j}x\})$ as a function of x for each county and describe what you see. Using maximum likelihood estimates only from those counties with 10 or more schools, obtain ad hoc estimates $\hat{\boldsymbol{\theta}}$ and $\hat{\Sigma}$ of $\boldsymbol{\theta}$ and Σ. Note that these estimates may not be representative of patterns from schools with small sample sizes.

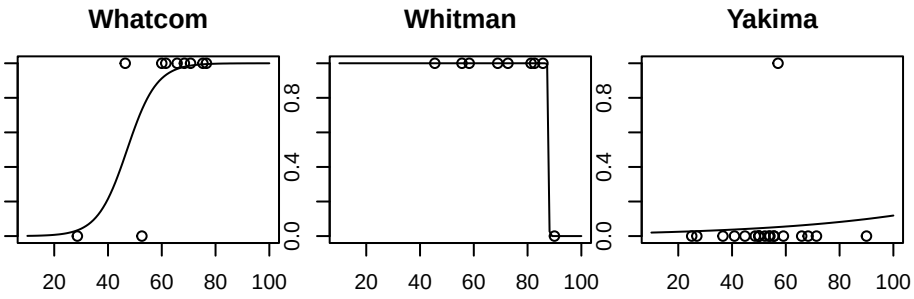
For counties with a small number of schools, the logistic regression tends to have large errors. And when the response variable of sampled data are all zeros or all ones, we can't find a proper regression result.

```
# Note: Data are preprocessed to replace space in the first column with underscore
mathstandard = as.data.frame(read.table("mathstandard.dat", header=TRUE))

# To see warnings
# options(warn=2, error=recover)
# resume default
# options(warn=0, error=NULL)

par(mfrow = c(3, 4), mar = c(1, 1, 3, 1))
# Create a county list to plot all counties
counties_list <- unique(mathstandard$county)

# plot for each county (not using ggplot)
for (county in counties_list)
{
county_data = mathstandard[which(mathstandard$county==county),]
model <- glm(county_data$metstandard ~ county_data$percentms, data = county_data, family=binomial)

# Create a sequence of values for the predictor variable
# x_seq <- seq(min(county_data$percentms), max(county_data$percentms), length.out = 100)
x_seq <- seq(10, 100, length.out = 100)

# Predict probabilities for each value in the sequence
pred_prob <- 1/(1+exp(-model$coef[2]*x_seq-model$coef[1]))

# Plot predicted probabilities against the predictor variable
plot(x_seq, pred_prob, type = "l", ylim = c(0, 1), xlab = "", ylab = "")
points(county_data$percentms, county_data$metstandard)
title(county)
```

```
}
```


Adams
Asotin
Benton
Chelan
Clallam
Clark
Columbia
Cowlitz
Douglas
Ferry
Franklin
Garfield
0.0
0.4
0.8
20
40
60
80
100


Grant
Grays_Harbor
Island
Jefferson
King
Kitsap
Kittitas
Klickitat
Lewis
Lincoln
Mason
Okanogan
Pacific
Pend_Oreille
Pierce
San_Juan
Skagit
Skamania
Snohomish
Spokane
Stevens
Thurston
Wahkiakum
Walla_Walla
0.0
0.4
0.8
20
40
60
80
100



We have 13 Warning messages when doing glm for each county:

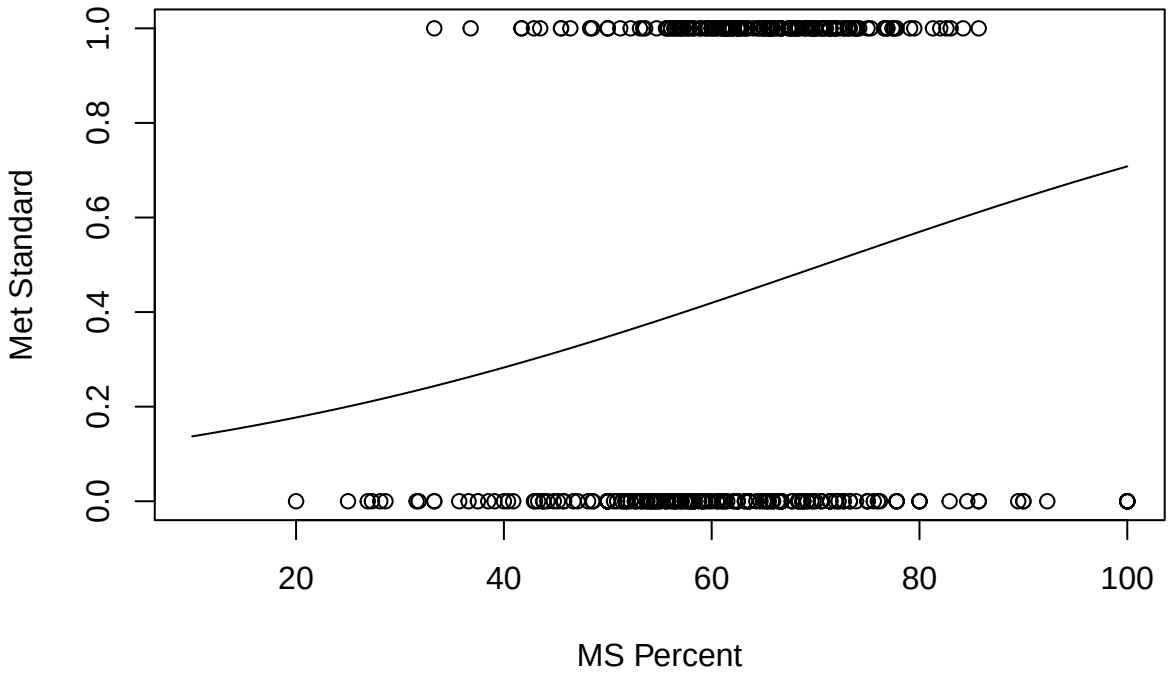
1. glm.fit: fitted probabilities numerically 0 or 1 occurred
2. glm.fit: fitted probabilities numerically 0 or 1 occurred
3. glm.fit: algorithm did not converge
4. glm.fit: fitted probabilities numerically 0 or 1 occurred
5. glm.fit: algorithm did not converge
6. glm.fit: fitted probabilities numerically 0 or 1 occurred
7. glm.fit: fitted probabilities numerically 0 or 1 occurred
8. glm.fit: fitted probabilities numerically 0 or 1 occurred
9. glm.fit: fitted probabilities numerically 0 or 1 occurred
10. glm.fit: fitted probabilities numerically 0 or 1 occurred
11. glm.fit: fitted probabilities numerically 0 or 1 occurred
12. glm.fit: algorithm did not converge
13. glm.fit: fitted probabilities numerically 0 or 1 occurred

These warnings comes from 13 counties that has data points ≤ 3

Counties like Columbia and Garfield has only 1 data point so there is no model fit for that group.

In counties like Spokane or Whitman, a few "abnormal" data point (high percentage of MS degrees but doesn't met standard) has too much leverage on the model fit that their model shows a opposite trend compared to the regression result for the entire state(which is shown in the next graph).

```r
model <- glm(mathstandard$metstandard ~ mathstandard$percentms, data = mathstandard, family=binomial)

# Create a sequence of values for the predictor variable
# x_seq <- seq(min(mathstandard$percentms), max(mathstandard$percentms), length.out = 100)
x_seq <- seq(10, 100, length.out = 100)

# Predict probabilities for each value in the sequence
pred_prob <- 1/(1+exp(-model$coef[2]*x_seq-model$coef[1]))

# Plot predicted probabilities against the predictor variable
plot(x_seq, pred_prob, type = "l", ylim = c(0, 1), xlab = "MS Percent", ylab = "Met Standard")
points(mathstandard$percentms, mathstandard$metstandard)
title("The Logistic Regression on the entire Washington State")
```




```
Ncounty <- table(mathstandard$county) # Table for number of sample data in each county
valid_counties <- names(which(Ncounty >= 10))

# Select counties with 10 or more schools

BETA <- NULL
for (county in valid_counties){
county_data = mathstandard[which(mathstandard$county==county),]
model <- glm(county_data$metstandard ~ county_data$percentms,
         data = county_data, family=binomial)
BETA<-rbind(BETA, model$coeff)
}

mu0<-thetahat<-apply(BETA, 2 , mean)
S0<-cov(BETA)
```

$$\hat{\boldsymbol{\theta}} = \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix} = \begin{pmatrix} -3.46 \\ 0.05 \end{pmatrix}$$

$$\hat{\boldsymbol{\Sigma}} = \begin{bmatrix} 10.756 & -0.179 \\ -0.1799 & 0.003 \end{bmatrix}$$

Part c

c) Formulate a unit information prior distribution for $\boldsymbol{\theta}$ and Σ based on the observed data. Specifically, let $\boldsymbol{\theta} \sim$ multivariate normal$(\hat{\boldsymbol{\theta}}, \hat{\Sigma})$ and let $\Sigma^{-1} \sim$ Wishart$(4, \hat{\Sigma}^{-1})$. Use a Metropolis-Hastings algorithm to approximate the joint posterior distribution of all parameters.

Our unit information prior of $\boldsymbol{\theta}$ is that it follows a multivariate normal distribution MVN $(\boldsymbol{\mu}, \boldsymbol{\Lambda}_0)$ with an expectation of $\boldsymbol{\mu} = \hat{\boldsymbol{\theta}} = \frac{1}{12}\sum_{j=1}^{12} \tilde{\boldsymbol{\beta}}_j$ and a prior covariance matrix $\boldsymbol{\Lambda}_0$ equal to $\hat{\boldsymbol{\Sigma}}$, the samples covariance matrix of $\tilde{\boldsymbol{\beta}}_j$.

Our unit information prior of $\boldsymbol{\Sigma}$ is that it follows a inverse-Wishart distribution $(\eta_0, \boldsymbol{S}_0)$. $\eta_0 = p + 2 = 4$ and $\boldsymbol{S}_0$ is also set equal to $\hat{\boldsymbol{\Sigma}}$.

Given Σ and our sample of regression coefficients $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_J$, the full conditional distribution of $\boldsymbol{\theta}$ is as follows:

$$\begin{aligned}\{\boldsymbol{\theta}|\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_J, \Sigma\} &\sim \text{multivariate normal}(\boldsymbol{\mu}_J, \Lambda_J), \text{ where} \\ \Lambda_J &= (\Lambda_0^{-1} + J\Sigma^{-1})^{-1} \\ \boldsymbol{\mu}_J &= \Lambda_J(\Lambda_0^{-1}\boldsymbol{\mu}_0 + J\Sigma^{-1}\overline{\boldsymbol{\beta}})\end{aligned}$$

where $\overline{\boldsymbol{\beta}}$ is the vector average $\frac{1}{J}\sum_{j=1}^{J} \boldsymbol{\beta}_j$.

The full conditional distribution of a covariance matrix is an inverse-Wishart distribution, with sum of squares matrix equal to the prior sum of sqares $\mathbf{S}_0$ plus the sum of squares from the sample:

$$\begin{aligned}\{\Sigma|\boldsymbol{\theta}, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_J\} &\sim \text{inverse-Wishart}(\eta_0 + J, [\mathbf{S}_0 + \mathbf{S}_\theta]^{-1}), \text{ where} \\ \mathbf{S}_\theta &= \sum_{j=1}^{J}(\boldsymbol{\beta}_j - \boldsymbol{\theta})(\boldsymbol{\beta}_j - \boldsymbol{\theta})^T\end{aligned}$$

For sampling $\boldsymbol{\beta}$, we follow these steps:

```
p<-2 # 1, percentms
J<-m<-39 # J and m are used interchangeably
eta0 <- p+2
# inverse Lambda_0, inverse S_0
iL0<-iSigma<-solve(S0)

# Recalculate BETA, the ith row represent beta for the ith county
# At the same time, prepare the data
BETA <- NULL
i<-1
Y<-data.frame(nrow=J)
X<-data.frame(nrow=J)
County.name<-data.frame(nrow=J)
for (county in counties_list){
county_data = mathstandard[which(mathstandard$county==county),]
if (nrow(county_data) <= 10){
    BETA <- rbind(BETA, c(0,0))
}else{
    model <- glm(county_data$metstandard ~ county_data$percentms,
             data = county_data, family=binomial)
    BETA<-rbind(BETA, model$coeff)
}
```

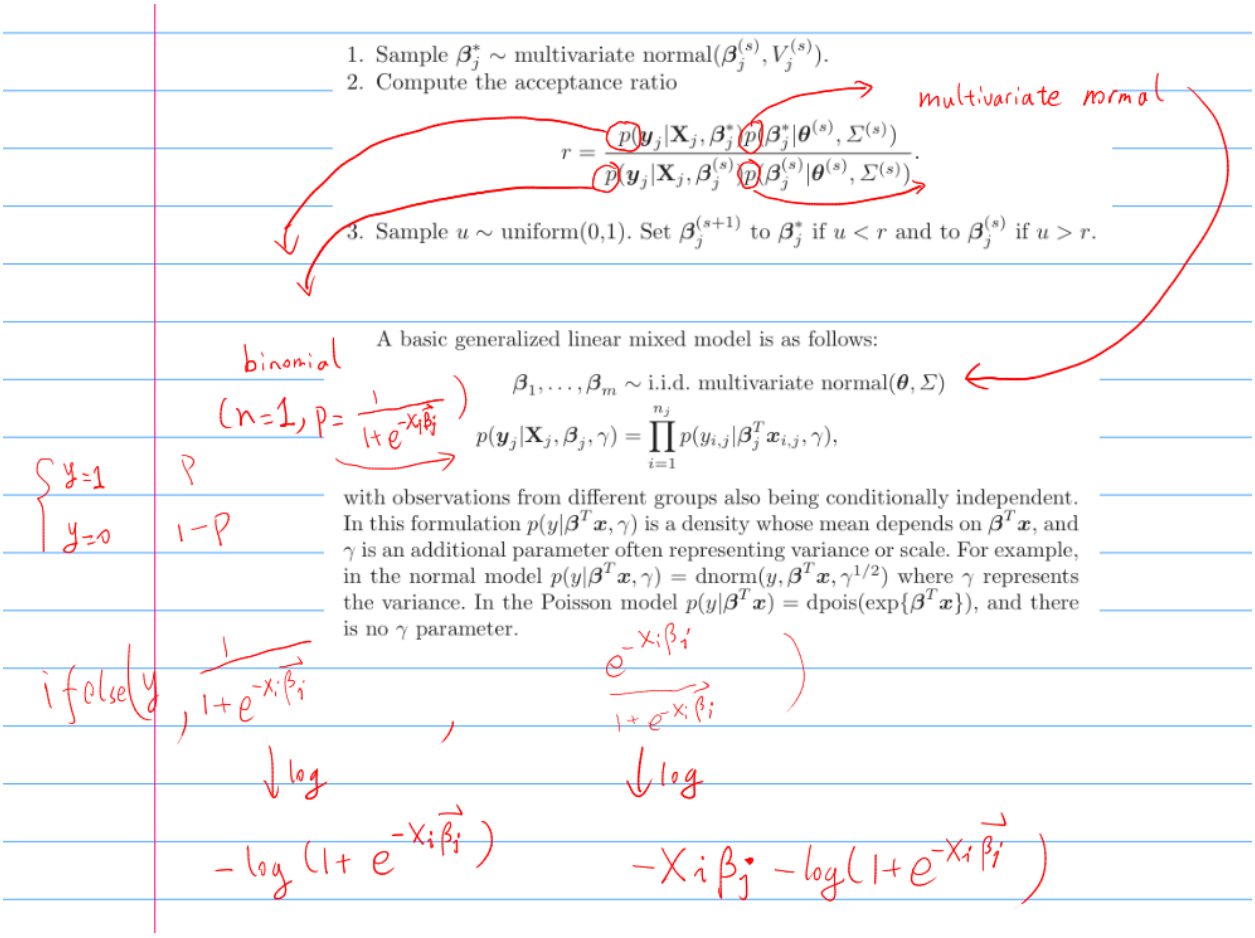


Figure 3: Sampling Beta using Metropolis-Hastings Algorithm (Sorry for the format...)

```
Y[i] <- t(county_data$metstandard)
X[i] <- t(county_data$percentms)
# Save the county name label
County.name[i] <- unique(county_data$county)
i<-i+1
}

# Utility function for sampling from binomial p(y|X,beta)
# to generate beta using Metropolis-Hastings Algorithm
XBeta <- function(X,Beta){
# X is a vector, Beta is a vector c(intercept, slop)
return(Beta[1]+Beta[2]*X)
}
sigmoid <- function(x){
    return(1/(1+exp(-x)))
}
# Alternative to dbinom, not working sadly...
loglikelihood <- function(X, beta, yy){
z<-XBeta(as.vector(t(X[j])),beta)
return(ifelse(yy,
  -log(1+exp(z)),
  -z-log(1+exp(z)))
)
}
## mvnormal simulation
rmvnorm<-function(n,mu,Sigma)
{
  E<-matrix(rnorm(n*length(mu)),n,length(mu))
  t(  t(E%*%chol(Sigma)) +c(mu))
}

## Wishart simulation
rwish<-function(n,nu0,S0)
{
  sS0 <- chol(S0)
  S<-array( dim=c( dim(S0),n ) )
  for(i in 1:n)
  {
     Z <- matrix(rnorm(nu0 * dim(S0)[1]), nu0, dim(S0)[1]) %*% sS0
     S[,,i]<- t(Z)%*%Z
  }
  S[,,1:n]
}
## mvnorm log density
ldmvnorm<-function(X,mu,Sigma,iSigma=solve(Sigma),dSigma=det(Sigma))
{
  Y<-t( t(X)-mu)
  sum(diag(-.5*t(Y)%*%Y%*%iSigma))  -
  .5*(  prod(dim(X))*log(2*pi) +     dim(X)[1]*log(dSigma) )
}
```

```r
## MCMC
ACCEPT.count <- rep(1,m)
THETA.post<-SIGMA.post<-NULL ; set.seed(1)
BETA.post <- list()
for(s in 1:500000)
{

  ##update theta
  Lm<-solve( iL0 +  m*iSigma )
  mum<-Lm%*%( iL0%*%mu0 + iSigma%*%apply(BETA,2,sum) )
  theta<-t(rmvnorm(1,mum,Lm))

  ##update Sigma
  mtheta<-matrix(theta,m,p,byrow=TRUE)
  iSigma<-rwish(1,eta0+m,
           solve( S0+t(BETA-mtheta)%*%(BETA-mtheta)) )

  ##update beta
  Sigma<-solve(iSigma) ; dSigma<-det(Sigma)
  for(j in 1:m)
  {
    # beta proposed (beta^(*)) from multivariate normal (beta_j^(s), V_J^(s)
    beta.p<-t(rmvnorm(1,BETA[j,],.5*Sigma))
########## Test dbinom ########
#yy <- as.vector(t(Y[1]))
#xx <- as.vector(t(X[1]))
#sigmoid(XBeta(xx,BETA[1,]))
#dbinom(yy, size=1, prob=sigmoid(XBeta(xx,BETA[1,])), log=TRUE)
##alternative code, not workign
#loglikelihood(xx, BETA[1], yy)
#dbinom(Y[j,],size=1, prob=exp(X%*%beta.p),log=TRUE )
#
#Or instead of using dbinom, we can use
#loglikelihood <- function(X, beta, yy){
#z<-XBeta(as.vector(t(X[j])),beta)
#return(ifelse(yy,
#  -log(1+exp(z)),
#  -z-log(1+exp(z))
#)
#}
########## Test dbinom ########
    lr<-sum(
        dbinom(as.vector(t(Y[j])),size=1,
           prob=sigmoid(XBeta(as.vector(t(X[j])),beta.p)),log=TRUE ) -
        dbinom(as.vector(t(Y[j])),size=1,
           prob=sigmoid(XBeta(as.vector(t(X[j])),BETA[j, ])),log=TRUE ) ) +
        ldmvnorm( t(beta.p),theta,Sigma,
              iSigma=iSigma,dSigma=dSigma ) -
        ldmvnorm( t(BETA[j,]),theta,Sigma,
              iSigma=iSigma,dSigma=dSigma )

    if( log(runif(1))<lr ) {
        ACCEPT.count[j] <- ACCEPT.count[j]+1
```

```
        BETA[j,]<-beta.p }
   }

  ##store some output
  if(s%%350==0) # saving every 350th value
  {
    cat(s,"\n")
    THETA.post<-rbind(THETA.post,t(theta))
    SIGMA.post<-rbind(SIGMA.post,c(Sigma))
    BETA.post[[length(BETA.post)+1]]<-BETA
  }

}
```

```
save(THETA.post, file="THETA.post.RData")
save(SIGMA.post, file="SIGMA.post.RData")
save(ACCEPT.count, file="ACCEPT.count.RData")
save(BETA, file="BETA.RData")
save(BETA.post, file="BETA.post.RData")
```

```
load("THETA.post.RData")
load("SIGMA.post.RData")
load("ACCEPT.count.RData")
load("BETA.RData")
load("BETA.post.RData")
```

Part d

d) Make plots of the samples of $\boldsymbol{\theta}$ and Σ (5 parameters) versus MCMC iteration number. Make sure you run the chain long enough so that your MCMC samples are likely to be a reasonable approximation to the posterior distribution.

```
## MCMC diagnostics
library(coda)
cat("The effective sample size of THETA")
```

```
## The effective sample size of THETA
```

```
round(apply(THETA.post,2,effectiveSize),2)
```

```
##           (Intercept) county_data$percentms
##               1282.46               1315.86
```

```
cat("The effective sample size of SIGMA")
```

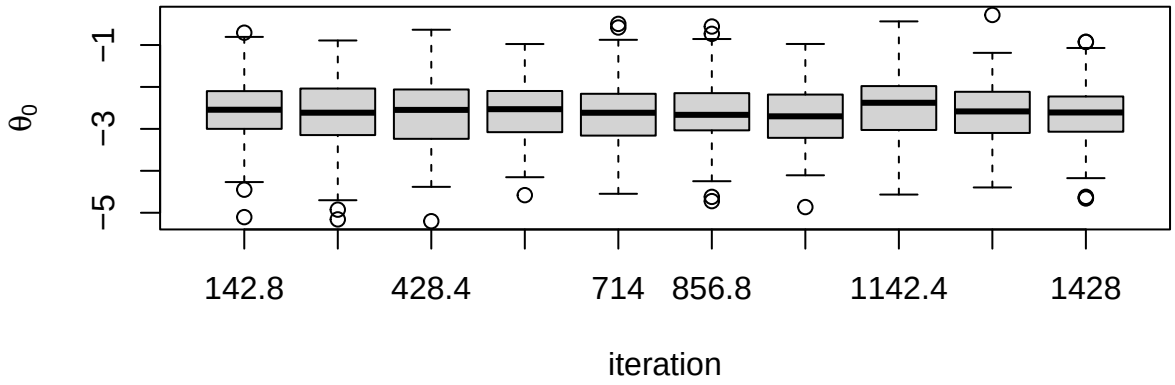
```
## The effective sample size of SIGMA
```

```
round(apply(SIGMA.post,2,effectiveSize),2)
```

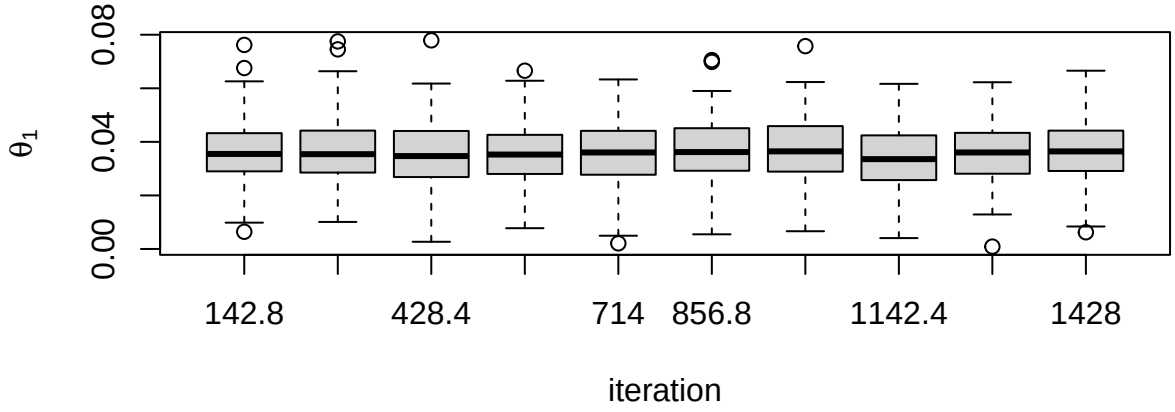
```
## [1] 1428 1428 1428 1428
```

Assess the convergence of the Markov chain

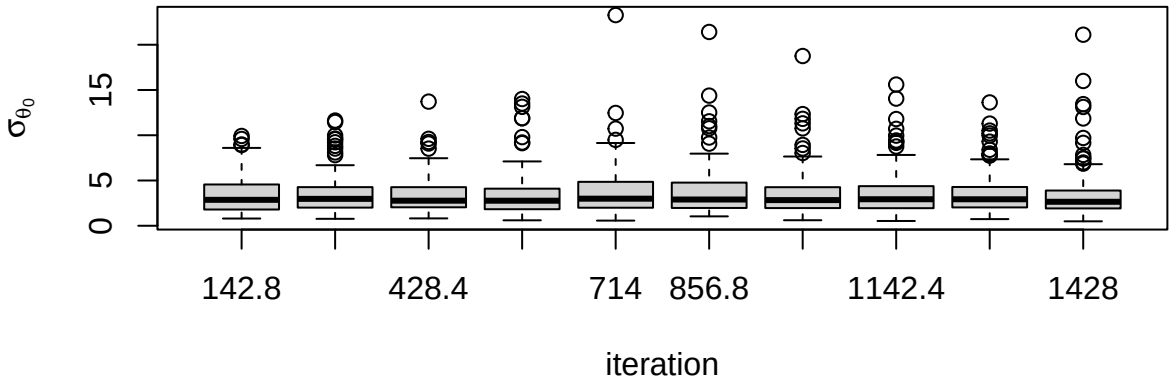
```
stationarity.plot(THETA.post[,1],xlab="iteration",ylab=expression(theta[0]))
```



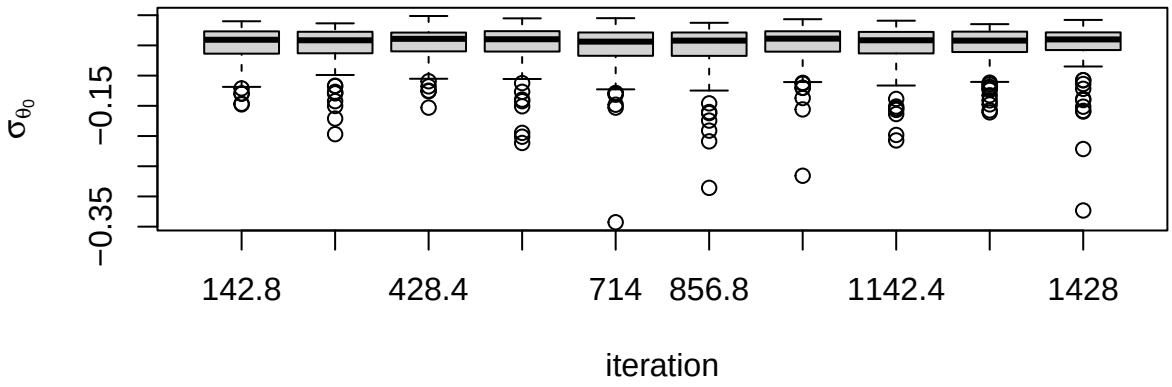

```
stationarity.plot(THETA.post[,2],xlab="iteration",ylab=expression(theta[1]))
```



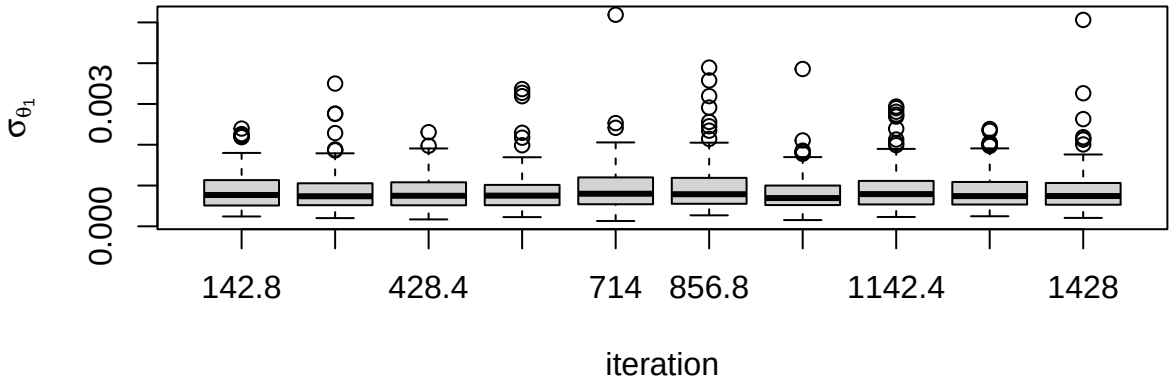

```
stationarity.plot(SIGMA.post[,1],xlab="iteration",ylab=expression(sigma[theta[0]]))
```



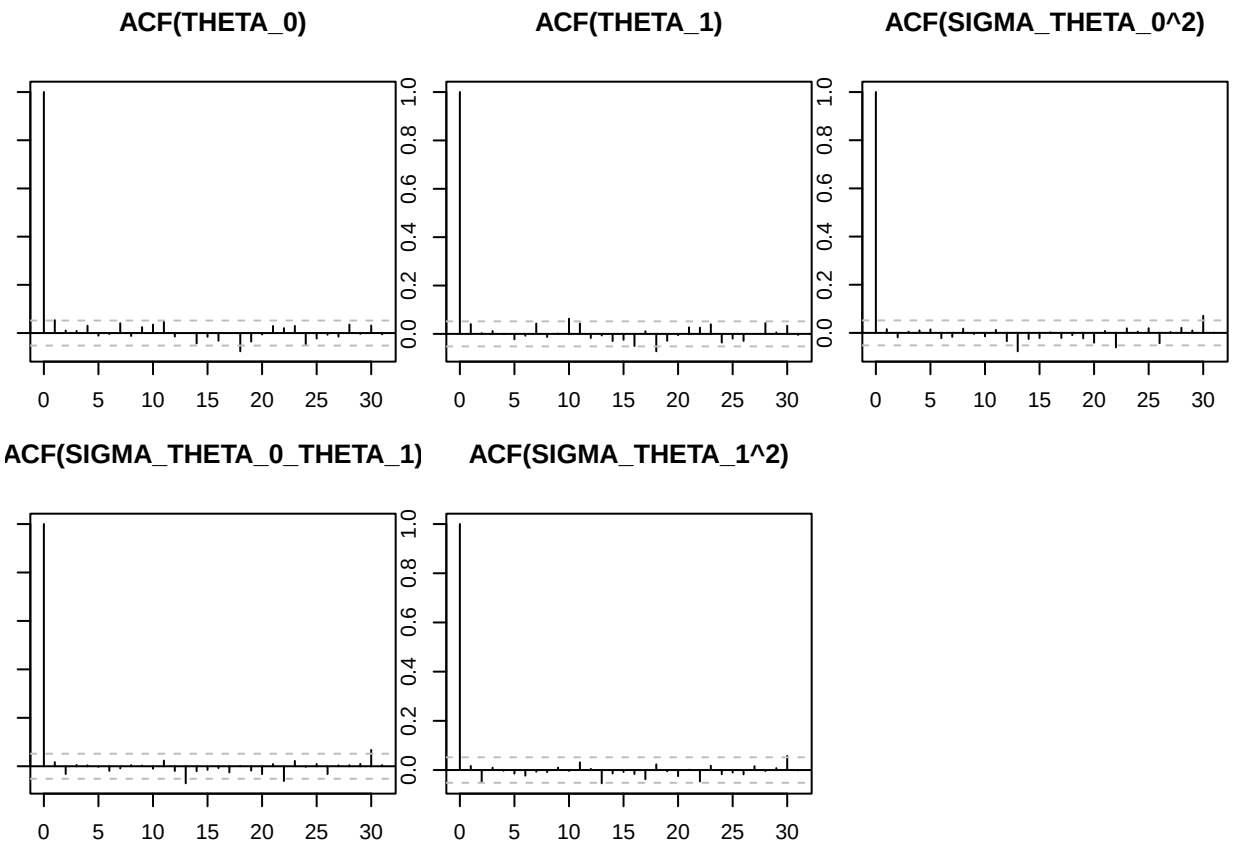

```
stationarity.plot(SIGMA.post[,2],xlab="iteration",ylab=expression(sigma[theta[0],theta[1]]))
```




```
stationarity.plot(SIGMA.post[,4],xlab="iteration",ylab=expression(sigma[theta[1]]))
```



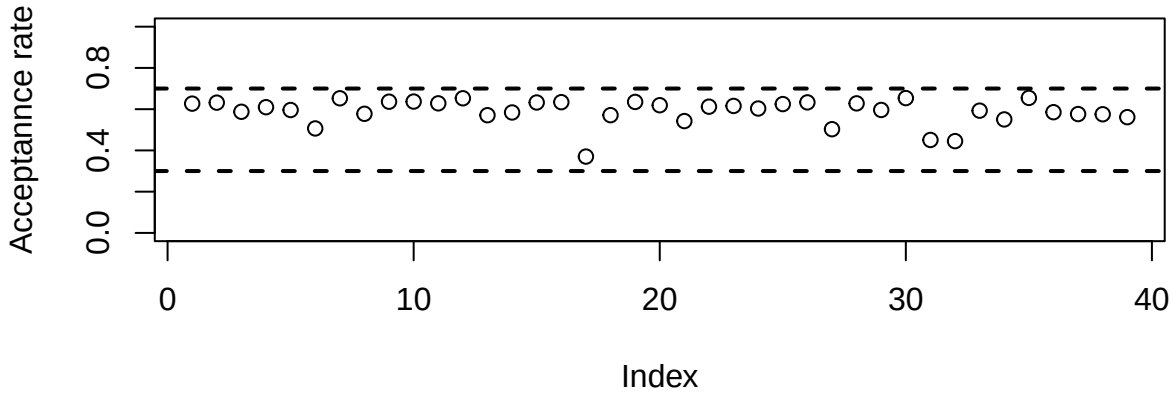

```
par(mfrow = c(2, 3), mar = c(3, 1, 3, 1))
acf(THETA.post[,1],ci.col="gray",xlab="lag", main = 'ACF(THETA_0)')
acf(THETA.post[,2],ci.col="gray",xlab="lag", main = 'ACF(THETA_1)')
acf(SIGMA.post[,1],ci.col="gray",xlab="lag", main = 'ACF(SIGMA_THETA_0^2)')
acf(SIGMA.post[,2],ci.col="gray",xlab="lag", main = 'ACF(SIGMA_THETA_0_THETA_1)')
acf(SIGMA.post[,4],ci.col="gray",xlab="lag", main = 'ACF(SIGMA_THETA_1^2)')
```



Assess the acceptance rate of Metropolis-Hastings Algorithm:

The acceptance rate when sampling $\boldsymbol{\beta}_j$ is plotted in this figure. The two dashed lines here are $r_1 = 0.3$ and $r_2 = 0.7$, we can see from this graph that our acceptance rate is a little bit high, but still acceptable.

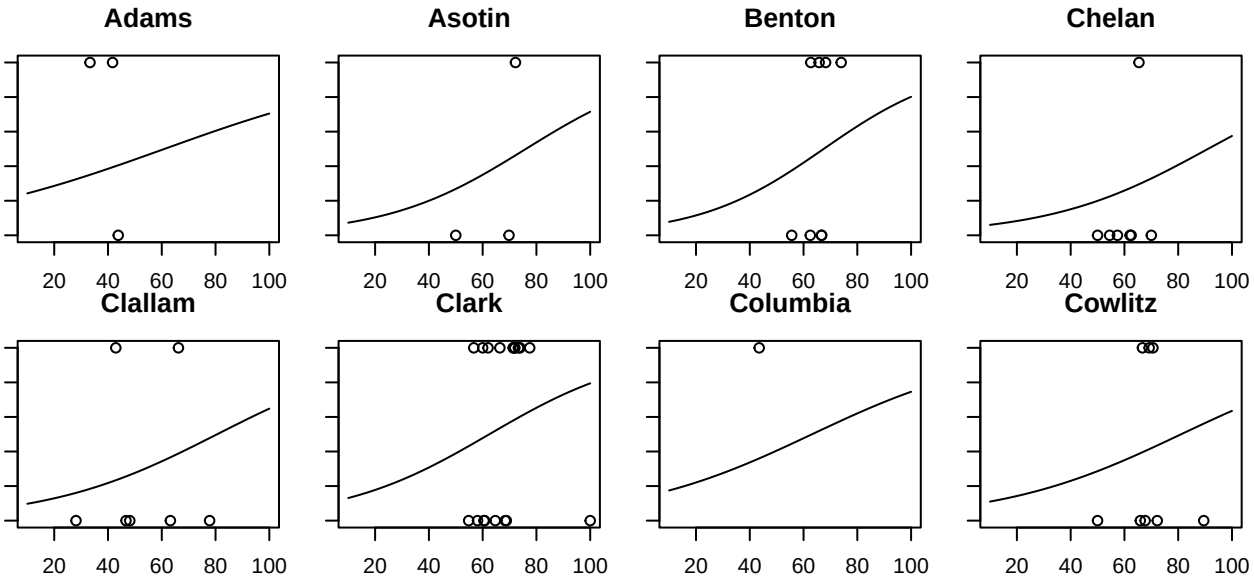
```
plot(ACCEPT.count/500000, ylim=c(0,1) , ylab="Acceptannce rate")
abline(h=0.3,lty=2,lwd=2)
abline(h=0.7,lty=2,lwd=2)
```

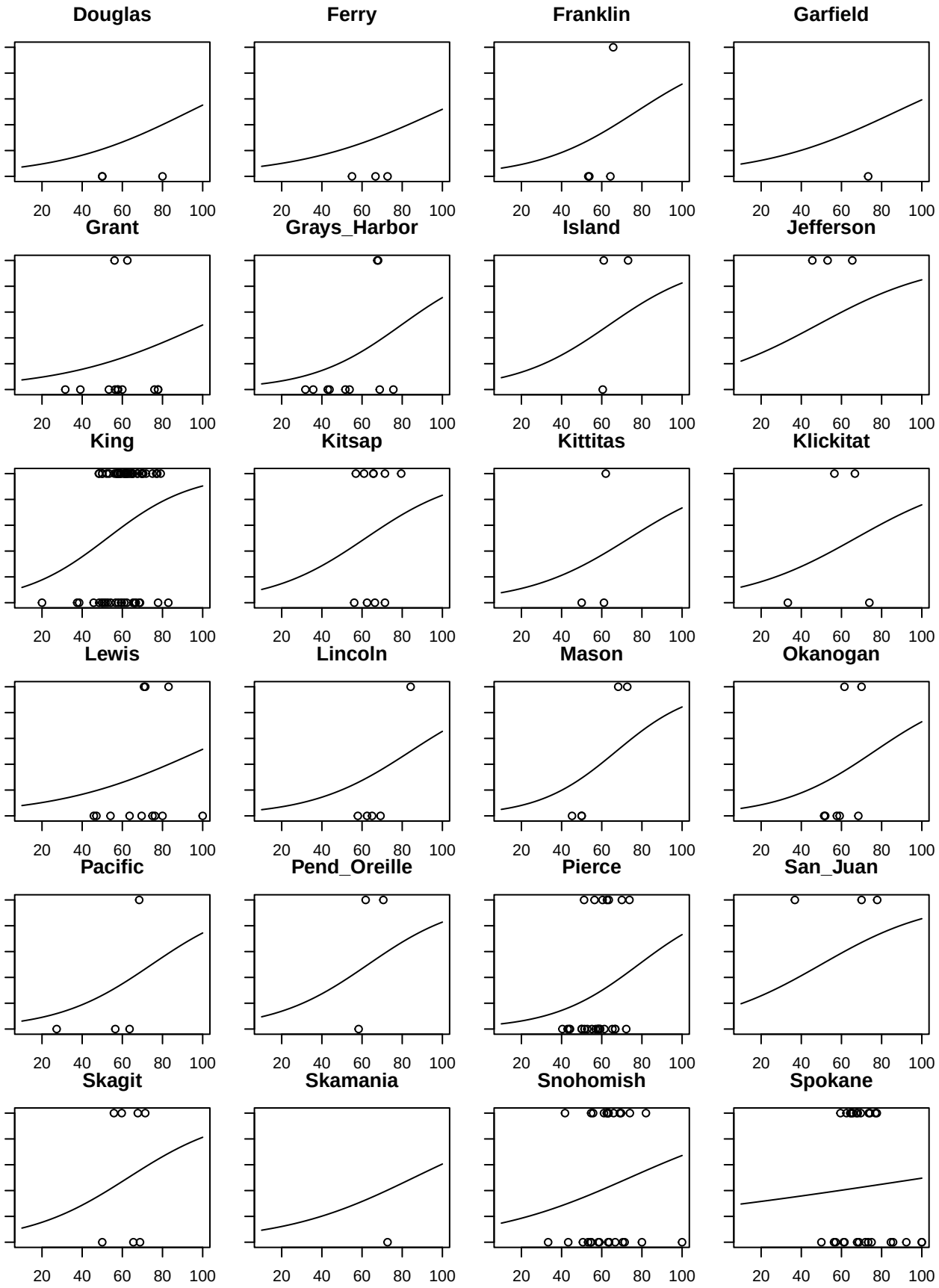


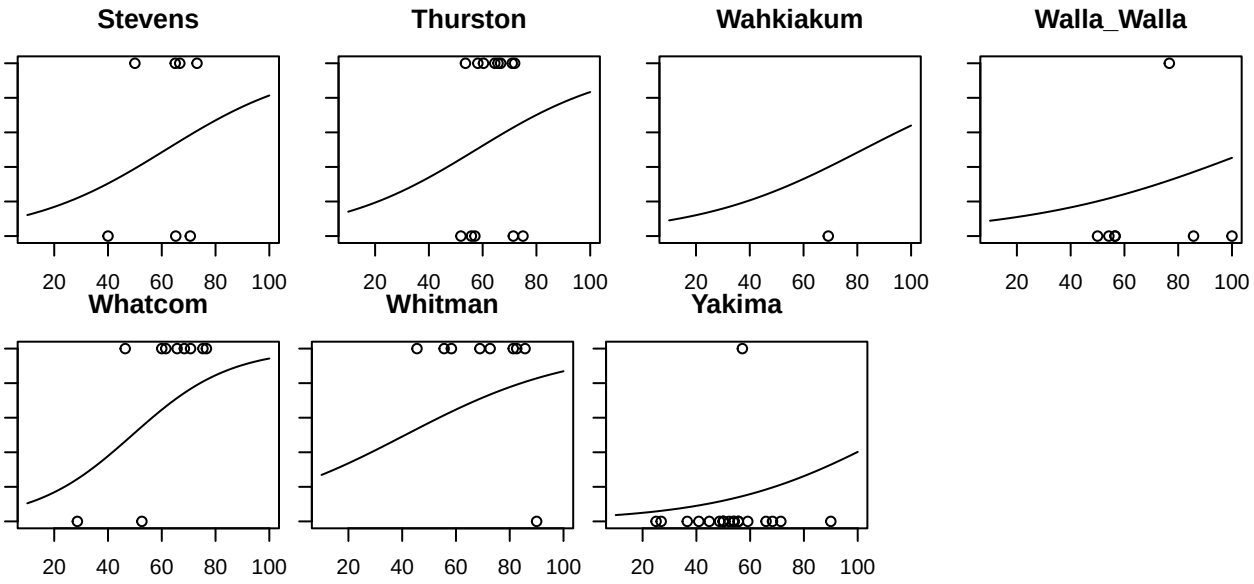
Part e

e) Obtain posterior expectations of $\boldsymbol{\beta}_j$ for each group j, plot $E[\beta_{0,j}|\mathbf{y}] + E[\beta_{1,j}|\mathbf{y}]x$ as a function of x for each county, compare to the plot in b) and describe why you see any differences between the two sets of regression lines.

```
# Traverse every county
j<-1
# Store the posterior expectations of beta
BETA_mean <- NULL
for (county in counties_list){
county_data = mathstandard[which(mathstandard$county==county),]

BETAj <- lapply(BETA.post, function(BETA) BETA[j, ])
# extract all posterior BETA_0 generated for county j
BETAj_0 <- unlist(lapply(BETAj, function(coeff) coeff[1]))
# extract all posterior BETA_1 generated for county j
BETAj_1 <- unlist(lapply(BETAj, function(coeff) coeff[2]))
BETA_mean <- rbind(BETA_mean,c(mean(BETAj_0),mean(BETAj_1)))

par(mfrow = c(3, 4), mar = c(1, 1, 3, 1))
x_seq <- seq(10, 100, length.out = 100)
pred_prob <- 1/(1+exp(-BETA_mean[j,2]*x_seq-BETA_mean[j,1]))
plot(x_seq, pred_prob, type = "l", ylim = c(0, 1), xlab = "", ylab = "")
points(county_data$percentms, county_data$metstandard)
title(county)
j<-j+1
}
```




Douglas
Ferry
Franklin
Garfield
Grant
Grays_Harbor
Island
Jefferson
King
Kitsap
Kittitas
Klickitat
Lewis
Lincoln
Mason
Okanogan
Pacific
Pend_Oreille
Pierce
San_Juan
Skagit
Skamania
Snohomish
Spokane
20 40 60 80 100

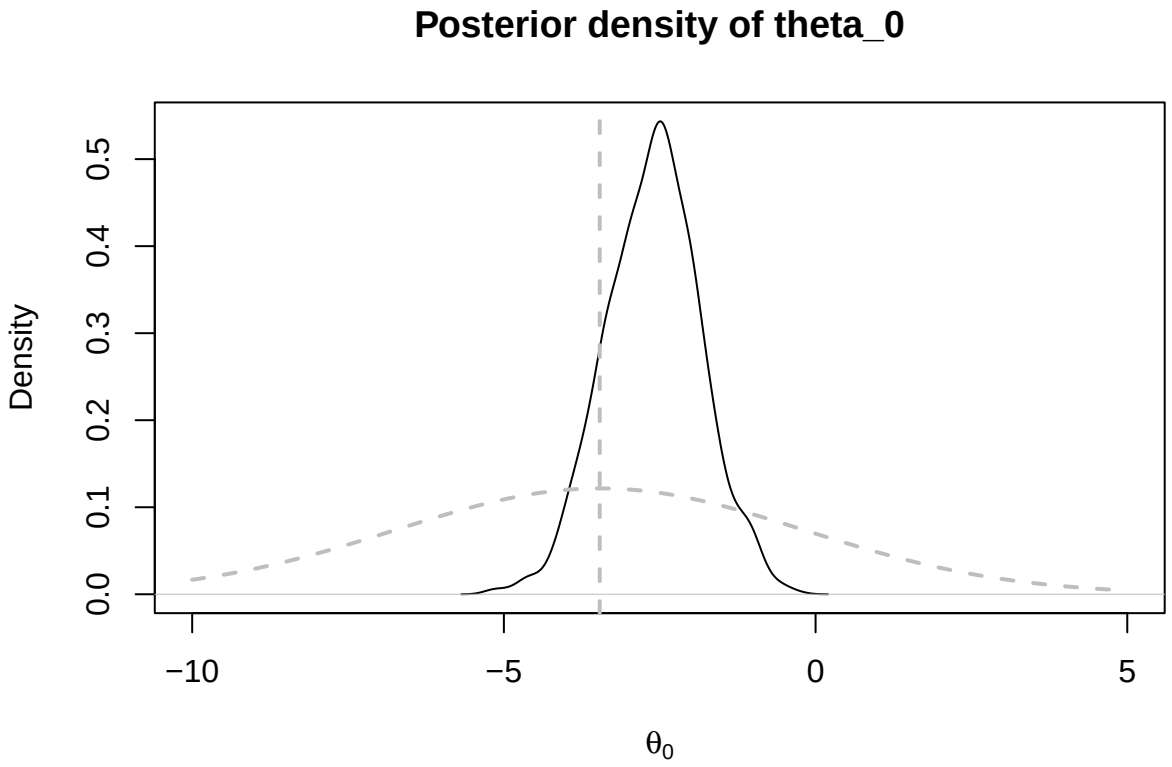


First of all, now all 39 counties has their models (as shown in above graphs). Secondly, we can see that even for counties like Whitman with some abnormal data points, we can still see a moderately increasing trend instead of an abrupt drop as shown in part b. This is because the hierarchical model for each county can borrow information from the whole population (data from entire Washington state). Furthermore, comparing counties of Whitman and Yakima, we can also see that our groupwise model reflects information collected from local (in each group) also. In this case, most schools in Whitman county met the standards whereas most schools in Yakima don't met the standard, so our model always predict a higher probability (regardless of their MS degree teacher proportions) for schools in Whitman to met the standard than that in Yakima.

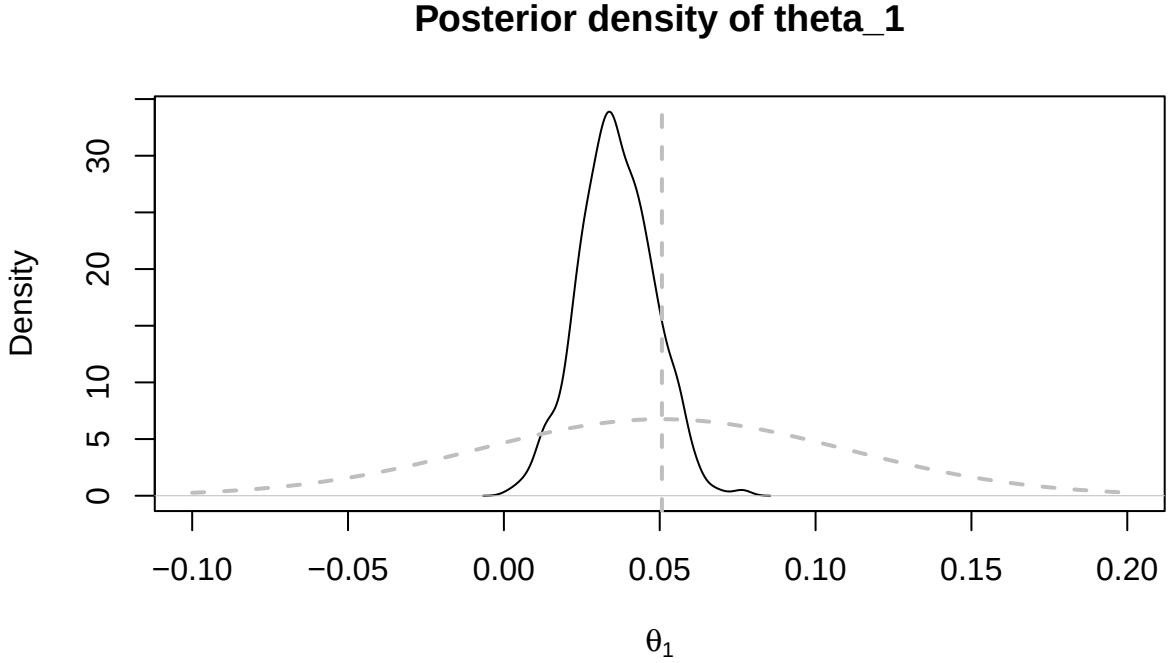
Part f

f) From your posterior samples, plot marginal posterior and prior densities of $\boldsymbol{\theta}$ and the elements of Σ. Include your ad hoc estimates from b) in the plots. Discuss the evidence that the slopes or intercepts vary across groups.

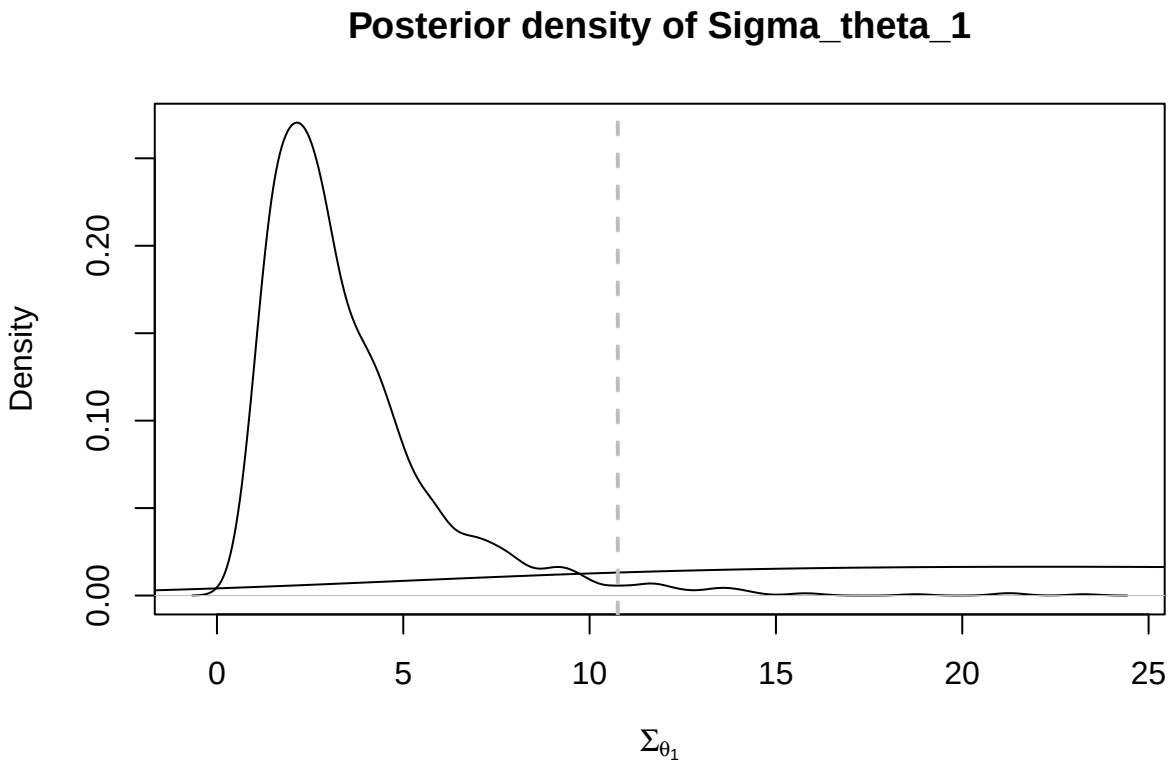
```
plot(density(THETA.post[,1]),xlim=c(-10,5), main="Posterior density of theta_0",
     xlab=expression(theta[0]))
x_theta <- seq(-10, 5, length.out=100)
lines(x_theta, dnorm(x_theta, mean=mu0[1], sd=sqrt(S0[1])), lty=2, col="GRAY", lwd=2)
abline(v=mu0[1], lty=2, col="GRAY", lwd=2)
```



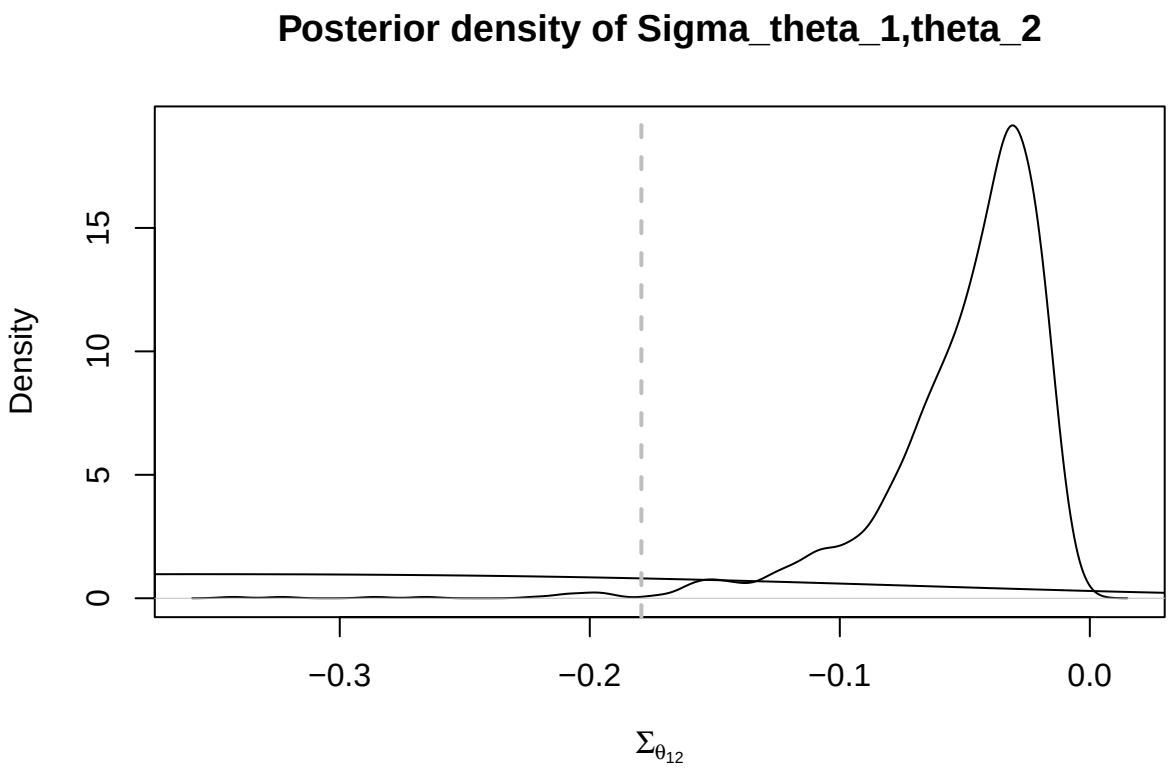

```
plot(density(THETA.post[,2]),xlim=c(-0.1,0.2), main="Posterior density of theta_1",
     xlab=expression(theta[1]))
x_theta <- seq(-0.1, 0.2, length.out=100)
lines(x_theta, dnorm(x_theta, mean=mu0[2], sd=sqrt(S0[4])), lty=2, col="GRAY", lwd=2)
abline(v=mu0[2], lty=2, col="GRAY", lwd=2)
```



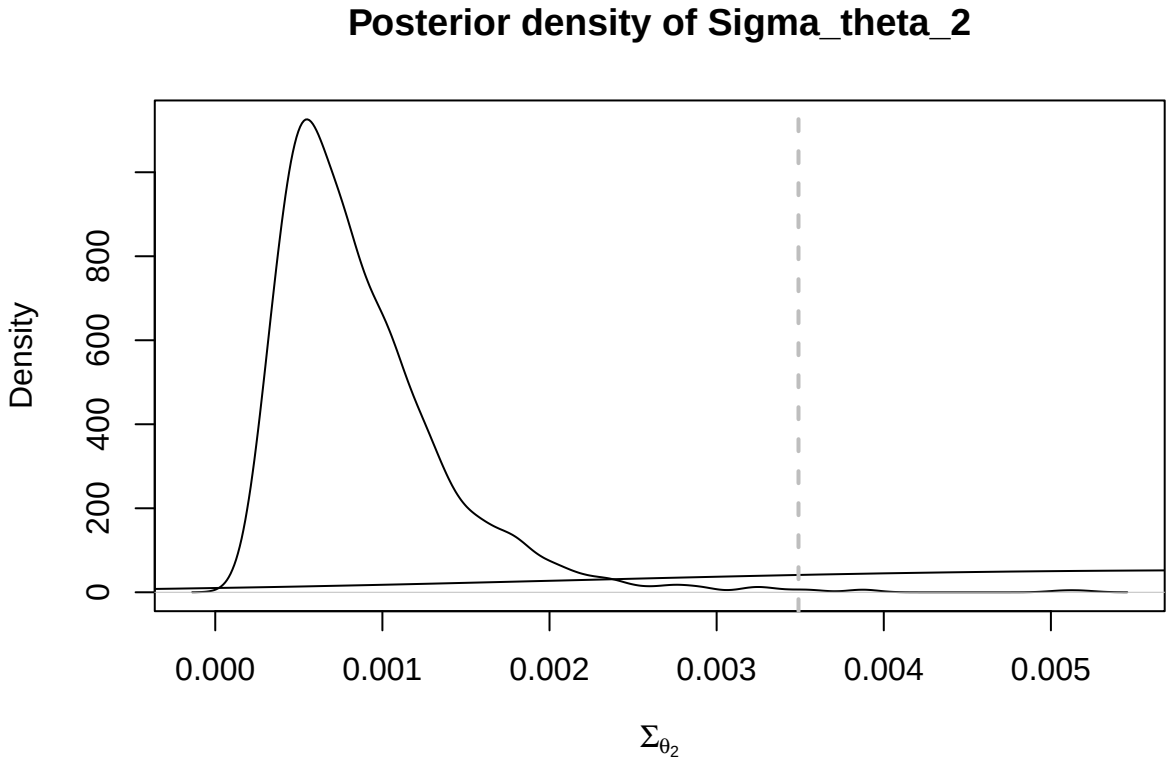
```
rwish<-function(n,nu0,S0)
{
  sS0 <- chol(S0)
  S<-array( dim=c( dim(S0),n ) )
  for(i in 1:n)
  {
     Z <- matrix(rnorm(nu0 * dim(S0)[1]), nu0, dim(S0)[1]) %*% sS0
     S[,,i]<- t(Z)%*%Z
  }
  S[,,1:n]
}
S0_prior <- rwish(n=1000, nu0=4, S0)
plot(density(SIGMA.post[,1]), main="Posterior density of Sigma_theta_1",
     xlab=expression(Sigma[theta[1]]))
lines(density(S0_prior[1,1,]))
abline(v=S0[1], lty=2, col="GRAY", lwd=2)
```




```
plot(density(SIGMA.post[,2]), main="Posterior density of Sigma_theta_1,theta_2",
     xlab=expression(Sigma[theta[12]]))
lines(density(S0_prior[1,2,]))
abline(v=S0[2], lty=2, col="GRAY", lwd=2)
```




```
plot(density(SIGMA.post[,4]), main="Posterior density of Sigma_theta_2",
     xlab=expression(Sigma[theta[2]]))
lines(density(S0_prior[2,2,]))
abline(v=S0[4], lty=2, col="GRAY", lwd=2)
```



The slope doesn't have much variance across groups, but the intercept varies more across groups. From these result we can infer that there are more unexplored factors (or inherent properties, such as socioeconomical status) in different counties other than the "percentms" that can explain the "metstandard" variable.