STA 360: Homework 9

Samuel Eure

11/9/2018

Problem 9.3

The Crime Data

```
set.seed(1)
data = read.table("crime.dat", header = T)
dataNames = names(data)[-1]
library(MASS);
Y.X <- data.matrix(data, rownames.force = NA)
X <- Y.X[,-1]
Y <- Y.X[,1]
```

Functions

```
getSSR_g <- function(X,Y,g){
  result <- t(Y)%*%(diag(length(X[,1]))-(g/(g+1))*X%*%solve(t(X)%*%X)%*%t(X))%*%Y;
}
getBetaOLS <- function(X,Y){
  return(solve(t(X)%*%X)%*%t(X)%*%Y);
}
getSSR <- function(B.ols,X,Y){
  return(t(Y)%*%Y - 2*t(B.ols)%*%t(X)%*%Y + t(B.ols)%*%t(X)%*%X%*%B.ols);
}
getSigma2OLS <- function(B.ols, X, Y){
  SSR = getSSR(B.ols,X,Y);
  s2ols = SSR/(length(X[,1]) - length(X[1,]));
  return(s2ols[1]);
}
getStandardErrors <- function(B, X, sigma2){
  xt.x.inverse <- solve(t(X)%*%X);
  return(sqrt(diag(xt.x.inverse)*sigma2));
}
getUnitInfoVariance<- function(B, X, sigma2, n){
  xt.x.inverse <- solve(t(X)%*%X);
  return(xt.x.inverse*sigma2*n);
}
```

```
g <- n <- 47; p <- 15;
nu0 = 2; s0.2 = 1;
beta.ols = getBetaOLS(X,Y);
sigma2.ols = getSigma2OLS(beta.ols, X, Y);
```

Monte Carlo Samples

```
Samples = 10000;
SSR.g = getSSR_g(X,Y,g);
SIGMA2 <- 1/rgamma(Samples, (nu0+n)/2, (nu0*sigma2.ols + SSR.g)/2);
xt.x.inverse <- solve(t(X)%*%X);
BETA <- matrix(0, nrow = length(SIGMA2), ncol = p);
for(i in 1:Samples){
  BETA[i,] <- mvrnorm(1, g*beta.ols/(g+1), g*SIGMA2[i]*xt.x.inverse/(g+1));
}
betaNames <- c()
for(k in 1:15){
  betaNames <- c(betaNames,paste("Beta", toString(k),": ", dataNames[k], sep = ""));
}

Summary = matrix(0, nrow = 15, ncol = 3)
colnames(Summary) = c("Mean", "2.5%", "97.5%")
rownames(Summary) = betaNames

for(k in 1:15){
  Summary[k, 1] <- mean(BETA[,k])
  Summary[k, 2:3] <- quantile(BETA[,k],c(.025, .975))
}
print("Posterior information")
```

```
## [1] "Posterior information"
```

```
print(Summary)
```

```
##                       Mean        2.5%       97.5%
## Beta1: M      2.797176e-01  0.05050938  0.51296186
## Beta2: So    -1.517898e-05 -0.30774699  0.30622456
## Beta3: Ed     5.329830e-01  0.23007109  0.82722670
## Beta4: Po1    1.451424e+00  0.05377408  2.84009734
## Beta5: Po2   -7.784345e-01 -2.24513658  0.66757015
## Beta6: LF    -6.557984e-02 -0.32644299  0.19617413
## Beta7: M.F    1.297843e-01 -0.13902793  0.39025129
## Beta8: Pop   -6.735417e-02 -0.27704278  0.14394989
## Beta9: NW     1.099525e-01 -0.18816115  0.39707669
## Beta10: U1   -2.659403e-01 -0.59191958  0.06758902
## Beta11: U2    3.600585e-01  0.04797487  0.66411662
## Beta12: GDP   2.350488e-01 -0.20337761  0.67915579
## Beta13: Ineq  7.098331e-01  0.30746319  1.10507045
## Beta14: Prob -2.807706e-01 -0.50927734 -0.05355299
## Beta15: Time -6.198556e-02 -0.27816930  0.15991672
```

```
#Least Squares
standErrors <- getStandardErrors(beta.ols, X, sigma2.ols)
SummaryOLS = matrix(0, nrow = 15, ncol = 3)
colnames(SummaryOLS) = c("OLS_Mean", "2.5%", "97.5%")
rownames(SummaryOLS) = betaNames
for(k in 1:15){
  SummaryOLS[k,1] <- beta.ols[k];
  SummaryOLS[k,2] <- beta.ols[k]-2*standErrors[k];
  SummaryOLS[k,3] <- beta.ols[k]+2*standErrors[k];
```

```
}
print("OLS")
```

```
## [1] "OLS"
```

```
print(SummaryOLS)
```

```
##                    OLS_Mean        2.5%       97.5%
## Beta1: M      0.2865177028  0.01919875  0.55383666
## Beta2: So    -0.0001179958 -0.36242447  0.36218848
## Beta3: Ed     0.5445161778  0.19095296  0.89807940
## Beta4: Po1    1.4716146465 -0.13594905  3.07917834
## Beta5: Po2   -0.7817757455 -2.45813939  0.89458790
## Beta6: LF    -0.0659672893 -0.36802793  0.23609335
## Beta7: M.F    0.1313002714 -0.17416954  0.43677009
## Beta8: Pop   -0.0702910179 -0.32013040  0.17954836
## Beta9: NW     0.1090590127 -0.22936245  0.44748048
## Beta10: U1   -0.2705407273 -0.65760644  0.11652499
## Beta11: U2    0.3687335028  0.01468306  0.72278395
## Beta12: GDP   0.2380580097 -0.27169225  0.74780827
## Beta13: Ineq  0.7262918200  0.26536648  1.18721716
## Beta14: Prob -0.2852262729 -0.54859472 -0.02185783
## Beta15: Time -0.0615771841 -0.31949841  0.19634404
```

Comparision of OLS and Bayesian Method

Generally speaking, the β vector obtained from the ordinary least squares method and the bayesian method look quite similar in terms of the expected values of $\beta_1, ..., \beta_{15}$. Moreover, the confidence intervals obtained by both methods look quite similar, however the bayesian method produces narrower confidence intervals for most of the beta values. In addition to this, my Bayesian intervals show that 0 is not in the confidence interval of Po_1, suggesting that it is a significant predictor of crime rates, however my ordinary least squares interval for Po_1 contains zero, suggesting otherwise.

Explanatory variables which seem to have the strongest relationship with crime rate are M(Percentage of males aged 14-24), Po1 (police expenditure in 1960), Ed (mean years of schooling), U2 (unemployment rate of urban males ages 35-39), Ineq (income inequality), and Prob (probability of imprisonment) with increased probability of imprisonment lowering the crime rate (on average) and increases in all of the other significant explanitory variables leading to an increase in crime rate. This is evidenced by the absense of zero in these explanatory variables' confidence intervals.

9.3b

i

```
#Dividing the data roughly in half
firstHalf <- sample(1:47, 23,replace = F, prob = rep(1/47,47));
secondHalf = c(setdiff(1:47, firstHalf));
testX <- X[firstHalf,];
testY <- Y[firstHalf];
trainX <- X[secondHalf,];
trainY <- Y[secondHalf];
```

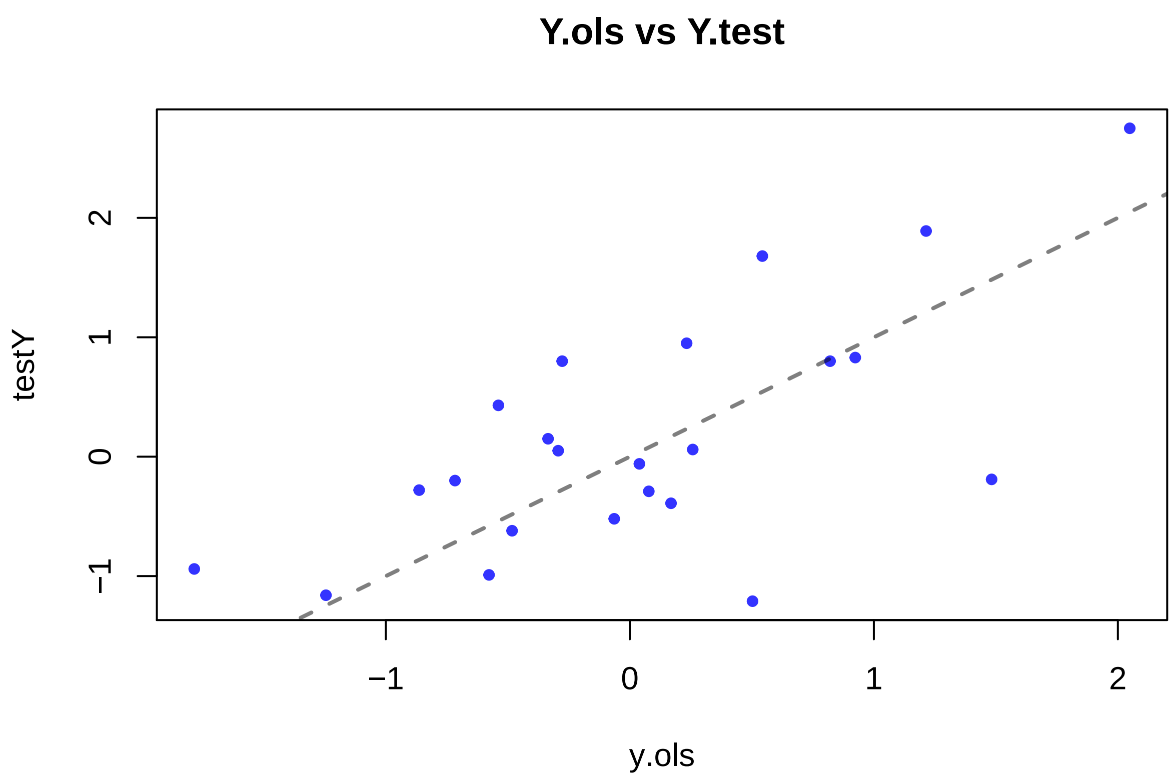
```r
trainBeta.ols <- getBetaOLS(trainX, trainY);
print("OLS Betas from Training Data")
```

```
## [1] "OLS Betas from Training Data"
```

```r
print(trainBeta.ols)
```

```
##             [,1]
## M      0.29513644
## So    -0.06115299
## Ed     0.38931920
## Po1    0.98165066
## Po2   -0.40682969
## LF     0.26909824
## M.F    0.08787996
## Pop   -0.04609962
## NW     0.49767913
## U1     0.06210397
## U2     0.16025710
## GDP    0.27653522
## Ineq   0.42708021
## Prob  -0.07273479
## Time   0.14060766
```

```r
y.ols <- testX%*%trainBeta.ols;
plot(y.ols, testY, col = rgb(0,0,1,.8), pch = 20, cex=1,
     main = "Y.ols vs Y.test")
abline(a = 0, b = 1, lty = 2, lwd = 2, col = rgb(0,0,0,.5))
```




```r
print("OLS Prediction Error");
```

```
## [1] "OLS Prediction Error"
```

```r
predError <- sum((y.ols - testY)^2)/length(testY);
print(predError)
```

```
## [1] 0.5726231
```

ii

```r
g = length(trainY);
trainSigma2.ols <- getSigma2OLS(trainBeta.ols, trainX, trainY);
trainSSR.g = getSSR_g(trainX,trainY,g);
trainSIGMA2 <- 1/rgamma(Samples, (nu0+length(trainY))/2, (nu0*trainSigma2.ols + trainSSR.g)/2);
train.xt.x.inverse <- solve(t(trainX)%*%trainX);
trainBETA <- matrix(0, nrow = length(trainSIGMA2), ncol = p);
for(i in 1:Samples){
  trainBETA[i,] <- mvrnorm(1, g*trainBeta.ols/(g+1),
                      g*trainSIGMA2[i]*train.xt.x.inverse/(g+1));
}
trainSummary = matrix(0, nrow = 15, ncol = 3)
colnames(trainSummary) = c("Mean", "2.5%", "97.5%")
rownames(trainSummary) = betaNames

for(k in 1:15){
  trainSummary[k, 1] <- mean(trainBETA[,k])
  trainSummary[k, 2:3] <- quantile(trainBETA[,k],c(.025, .975))
}
print("Trained Beta Values Summary")
```

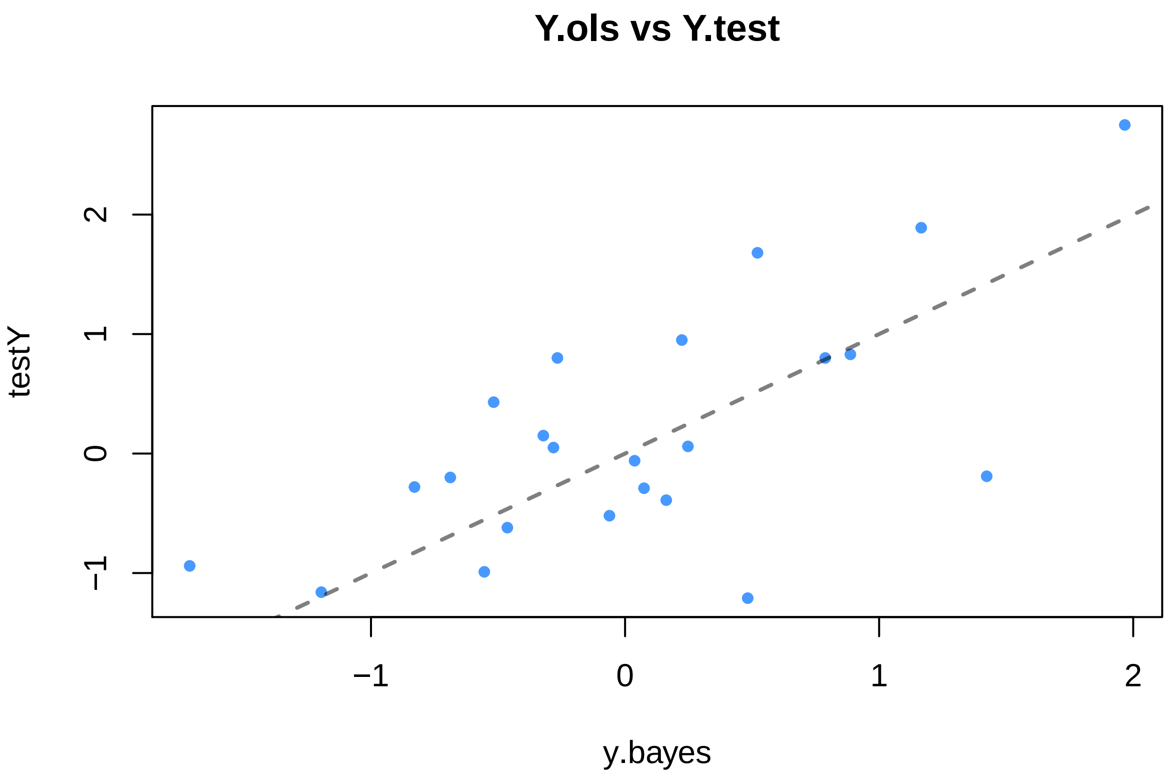
```
## [1] "Trained Beta Values Summary"
```

```r
print(trainSummary)
```

```
##                     Mean       2.5%     97.5%
## Beta1: M      0.27939040 -0.2754566 0.8434932
## Beta2: So    -0.05660868 -0.9116444 0.8375489
## Beta3: Ed     0.37720489 -0.3258930 1.0650089
## Beta4: Po1    0.93895602 -1.0905417 2.9628649
## Beta5: Po2   -0.38893861 -2.6395720 1.8620593
## Beta6: LF     0.25480780 -0.4385561 0.9651008
## Beta7: M.F    0.08590832 -0.3173931 0.4883991
## Beta8: Pop   -0.04346307 -0.4710220 0.3802389
## Beta9: NW     0.47750004 -0.4324989 1.4154862
## Beta10: U1    0.05993442 -0.5354867 0.6504630
## Beta11: U2    0.15300117 -0.3720993 0.6841563
## Beta12: GDP   0.27393239 -0.7878848 1.3056244
## Beta13: Ineq  0.41625927 -0.6109912 1.4145682
## Beta14: Prob -0.06608849 -0.7237684 0.6005636
## Beta15: Time  0.13478909 -0.2507081 0.5295180
```

```r
#Given the g prior
bayes.beta <- trainBeta.ols*g/(g+1)
y.bayes <- testX%*%bayes.beta;
plot(y.bayes, testY, col = rgb(.1,.5,1,.8), pch = 20, cex=1,
     main = "Y.ols vs Y.test")
```

```r
abline(a = 0, b = 1, lty = 2, lwd = 2, col = rgb(0,0,0,.5))
```




```r
print("Bayes Prediction Error");
```

```
## [1] "Bayes Prediction Error"
```

```r
bayes.predError <- sum((y.bayes - testY)^2)/length(testY);
print(bayes.predError)
```

```
## [1] 0.5605371
```

My prediction error using the bayesian method is slightly lower (about 0.01) lower than my ordinary least squares error. This makes sense given that $\nu_0 = 2$ is quite small and $\frac{47}{48} \approx 1$.

9.3c

For Ordinary Least Squares

```r
Runs = 1000;
GROUP_1 <- matrix(0, nrow = Runs, ncol = 23);
GROUP_2 <- matrix(0, nrow = Runs, ncol = 24);
OLS_BETAS  <- matrix(0, nrow = Runs, ncol = 15);
PRED_ERROR <- matrix(0, nrow = Runs, ncol = 1);
for(run in 1:Runs){
  GROUP_1[run,] <- sample(1:47, 23,replace = F);
  GROUP_2[run,] <- c(setdiff(1:47, GROUP_1[run,]))
}
for(k in 1:Runs){
  OLS_BETAS[k,]   <- getBetaOLS(X[GROUP_1[k,],], Y[GROUP_1[k,]])
```

```r
  prediction      <- X[GROUP_2[k,],]%*%OLS_BETAS[k,]
  PRED_ERROR[k,] <- sum((prediction - Y[GROUP_2[k,]])^2/length(GROUP_2[1,]))
}
ols.average = mean(PRED_ERROR[,1]);
print("Expected Value given OLS");
```

```
## [1] "Expected Value given OLS"
```

```r
print(ols.average);
```

```
## [1] 1.00169
```

```r
print("Confidence interval");
```

```
## [1] "Confidence interval"
```

```r
CI95 = matrix(0, nrow = 1, ncol = 2);
CI95[1,] <- quantile(PRED_ERROR, c(0.025, 0.975));
colnames(CI95) <- c("2.5%", "97.5%");
print(CI95)
```

```
##           2.5%    97.5%
## [1,] 0.4292764 2.514804
```

For The Bayesian Approach

```r
g = 23;
SmallerSamples <- 1000;
BAYES_ERROR    <- matrix(0, nrow = SmallerSamples, ncol = 1);
for(run in 1:Runs){
  localTrainX  <- X[GROUP_1[run,],];
  localTrainY  <- Y[GROUP_1[run,]];
  localTestX  <- X[GROUP_2[run,],];
  localTestY  <- Y[GROUP_2[run,]];
  localOlsBeta <- OLS_BETAS[run,];
  localBeta.bayes <- localOlsBeta*g/(g+1);
  pred.bayes <- localTestX%*%localBeta.bayes
  BAYES_ERROR[run,1] <- sum((pred.bayes - localTestY)^2/length(GROUP_2[1,]))
}
print("Expected Error using Bayesian Method")
```

```
## [1] "Expected Error using Bayesian Method"
```

```r
print(mean(BAYES_ERROR))
```

```
## [1] 0.947454
```

```r
CIBayes95     <- matrix(0, nrow = 1, ncol = 2);
CIBayes95[1,] <- quantile(BAYES_ERROR, c(0.025, 0.975))
print("With CI")
```

```
## [1] "With CI"
```

```r
colnames(CIBayes95) <- c("2.5%", "97.5%");
print(CIBayes95)
```

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

```
## [1,] 0.415769 2.33486
```

Hierarchical Modeling

Part 1: The Derivation of Gibbs

Using priors of

$$\beta_0 \sim MVN(\mu, \Lambda)$$
$$\Sigma_0^{-1} \sim Wishart(\eta_0, S_0^{-1})$$

and for each school j, sampling

$$\beta_j \sim MVN(\beta_0, \Sigma_0)$$

and approximating $Y_{i,j}$ (the score of the i^{th} student in the j^{th} school) using the model

$$Y_{i,j} \sim N(\beta_j^T X_{i,j}, \sigma^2)$$

which can be represented as

$$Y_{1:n_j,j} \sim MVN(X_{1:n_j,j}\beta_j, \sigma^2 I)$$

where $Y_{1:n_j,j} \in \mathbb{R}^{n_j x 1}$ is vector of the scores of each student in school j, and $X_{1:n_j,j} \in \mathbb{R}^{1:n_j \chi p}$ is the matrix where the k^{th} row represent the k^{th} student (one of n_j students in school j), of school j, and each column represents a different covariate (one of p) for that student. The $\sigma^2 I$ represents the identity matrix in $I \in \mathbb{R}^{n_j \chi n_j}$. From now on, I shall represent $X_{1:n_j,j}$ simply as X_j.

the full conditional distributions of our unknown values can be calculated through the joint distribution

$$p(Y, X, \beta_{1:m}, \beta_0, \sigma^2, \Sigma_0) = \left[\prod_{j=1}^{m} p(Y_j \mid \beta_j, X_j, \sigma^2 I) p(\beta_j \mid \beta_0, \Sigma_0)\right] p(\sigma^2) p(\beta_0) p(\Sigma_0^{-1})$$

Full Conditional of β_j

$$p(\beta_j \mid \dots) \propto p(Y, X, \beta_{1:m}, \beta_0, \sigma^2, \Sigma_0) \propto p(Y_j \mid \beta_j, X_j, \sigma^2 I) p(\beta_j \mid \beta_0, \Sigma_0)$$
$$\propto \left(e^{-\frac{1}{2}(\beta_j - \beta_0)^T \Sigma_0^{-1} (\beta_j - \beta_0)}\right)\left(e^{-\frac{1}{2\sigma^2}(Y_j - X_j\beta_j)^T (Y_j - X_j\beta_j)}\right) \propto e^{-\frac{1}{2}\beta_j^T \Sigma_0^{-1}\beta_j + \beta_j^T \Sigma_0^{-1}\beta_0} e^{-\frac{1}{2}\beta_j^T \frac{X_j^T X_j}{\sigma^2}\beta_j^T + \beta_j^T \frac{X_j^T Y_j}{\sigma^2}}$$

$$\propto e^{-\frac{1}{2}\beta_j^T\left(\Sigma_0^{-1} + X_j^T X_j/\sigma^2\right)\beta_j^T + \beta_j^T\left(\Sigma_0^{-1}\beta_0 + X_j^T Y_j/\sigma^2\right)}$$

$$\implies (\beta_j \mid \dots) \sim \text{MVN}\left(\left[\Sigma_0^{-1} + X_j^T X_j/\sigma^2\right]^{-1}\left[\Sigma_0^{-1}\beta_0 + X_j^T Y_j/\sigma^2\right], \left[\Sigma_0^{-1} + X_j^T X_j/\sigma^2\right]^{-1}\right)$$

Full Conditional of β_0

$$p(\beta_0 \mid \dots) \propto \left[\prod_{j=1}^{m} p(\beta_j \mid \beta_0, \Sigma_0)\right] p(\beta_0)$$

$$\propto \left[e^{-\frac{1}{2}\sum_{j=1}^{m}(\beta_j-\beta_0)^T\Sigma_0^{-1}(\beta_j-\beta_0)}\right] e^{-\frac{1}{2}(\beta_0-\mu)^T\Lambda^{-1}(\beta_0-\mu)} \propto \left(e^{-\frac{1}{2}m\beta_0^T\Sigma_0^{-1}\beta_0+\beta_0^T m\Sigma_0^{-1}\bar{\beta}}\right)\left(e^{-\frac{1}{2}\beta_0^T\Lambda^{-1}\beta_0+\beta_0\Lambda^{-1}\mu}\right)$$

$$\propto e^{-\frac{1}{2}\beta_0^T\left[\Lambda^{-1}+m\Sigma_0^{-1}\right]\beta_0+\beta_0^T\left[\Lambda^{-1}\mu+m\Sigma_0^{-1}\bar{\beta}\right]}, \quad \bar{\beta} = \frac{1}{m}\sum_{j=1}^{m}\beta_j$$

$$\implies (\beta_0 \mid \dots) \sim MVN(\left[\Lambda^{-1}+m\Sigma_0^{-1}\right]^{-1}\left[\Lambda^{-1}\mu+m\Sigma_0^{-1}\bar{\beta}\right], \left[\Lambda^{-1}+m\Sigma_0^{-1}\right]^{-1})$$

Full Conditional for σ^2

$$p(1/\sigma^2 \mid \dots) \propto \left[\prod_{j=1}^{m} p(Y_j \mid \beta_j, X_j, \sigma^2 I)\right] p(1/\sigma^2)$$

$$\propto \left(\left|I\right|^{-\frac{\sum_{j=1}^{m} n_j}{2}}(1/\sigma^2)^{\frac{\sum_{j=1}^{m} n_j}{2}} e^{-\frac{1}{\sigma^2}\sum_{j=1}^{m}(Y_j-X_j\beta_j)^T(Y_j-X_j\beta_j)}\right)\left((1/\sigma^2)^{\frac{\nu_0}{2}-1}e^{-\frac{\nu_0\sigma_0^2}{2}\frac{1}{\sigma^2}}\right)$$

$$\propto \left(1/\sigma^2\right)^{\frac{\nu_0+\sum_{j=1}^{m} n_j}{2}-1} e^{-\left(\frac{\nu_0\sigma_0^2+\sum_{j=1}^{m}(Y_j-X_j\beta_j)^T(Y_j-X_j\beta_j)}{2}\right)\frac{1}{\sigma^2}}$$

$$\implies (\sigma^2 \mid \dots) \sim \text{Inv-Gamma}\left(\frac{\nu_0+\sum_{j=1}^{m} n_j}{2}, \frac{\nu_0\sigma_0^2+\sum_{j=1}^{m}(Y_j-X_j\beta_j)^T(Y_j-X_j\beta_j)}{2}\right)$$

Full Conditional for Σ_0^{-1}

$$p(\Sigma_0^{-1} \mid \dots) \propto \left[\prod_{j=1}^{m} p(\beta_j \mid \beta_0, \Sigma_0)\right] p(\Sigma_0^{-1})$$

$$\propto \left(\left|\Sigma_0^{-1}\right|^{\frac{m}{2}} e^{-\frac{1}{2}\sum_{j=1}^{m}(\beta_j-\beta_0)^T\Sigma_0^{-1}(\beta_j-\beta_0)}\right)\left(\left|\Sigma_0^{-1}\right|^{\frac{\eta_0-p-1}{2}} e^{-\frac{1}{2}tr(S_0\Sigma_0^{-1})}\right) \propto \left|\Sigma_0^{-1}\right|^{\frac{(\eta_0+m)-p-1}{2}} e^{-\frac{1}{2}tr\left(\left[S_0+S_\beta\right]\Sigma_0^{-1}\right)}$$

$$S_\beta = \sum_{j=1}^{m}(\beta_j-\beta_0)(\beta_j-\beta_0)^T$$

$$\implies (\Sigma_0^{-1} \mid \dots) \sim \text{Wishart}(\eta_0+m, \left[S_0+S_\beta\right]^{-1})$$

Gibbs

Starting with a gift from an oricle of initial values of $\left(\sigma^2\right)^{(0)}$, $\left(\Sigma_0^{-1}\right)^{(0)}$, and $\beta_0^{(0)}$

(1) For j in $1:m$

$$\to \beta_j^{(s+1)} \sim \text{MVN}\left(\left[\left(\Sigma_0^{-1}\right)^{(s)}+X_j^T X_j/\left(\sigma^2\right)^{(s)}\right]^{-1}\left[\left(\Sigma_0^{-1}\right)^{(s)}\beta_0^{(s)}+X_j^T Y_j/\left(\sigma^2\right)^{(s)}\right], \left[\left(\Sigma_0^{-1}\right)^{(s)}+X_j^T X_j/\left(\sigma^2\right)^{(s)}\right]^{-1}\right)$$

(2)

$$\rightarrow (\sigma^2)^{(s+1)} \sim \text{Inv-Gamma}\Bigg(\frac{\nu_0 + \sum_{j=1}^m n_j}{2}, \frac{\nu_0\sigma_0^2 + \sum_{j=1}^m (Y_j - X_j\beta_j^{(s+1)})^T(Y_j - X_j\beta_j^{(s+1)})}{2}\Bigg)$$

(3)

$$\rightarrow \beta_0^{(s+1)} \sim MVN(\big[\Lambda^{-1} + m\big(\Sigma_0^{-1}\big)^{(s)}\big]^{-1}\big[\Lambda^{-1}\mu + m\big(\Sigma_0^{-1}\big)^{(s)}\big(\bar{\beta}\big)^{(s+1)}\big], \big[\Lambda^{-1} + m\big(\Sigma_0^{-1}\big)^{(s)}\big]^{-1})$$

(4)

$$\rightarrow (\Sigma_0^{-1})^{(s+1)} \sim \text{Wishart}(\eta_0 + m, \big[S_0 + S_\beta^{(s+1)}\big]^{-1})$$

Part II: The Implementation

Functions

```
library(MASS)
generateX <- function(Nk, p){
  Covariance <- matrix(.2, nrow = p, ncol = p) + diag(p)*.8;
  X <- matrix(0, nrow = 1, ncol = p);
  for(i in 1:Nk){
    Row_i <- mvrnorm(1,rep(0, p), Covariance);
    X <- rbind(X, Row_i);
  }
  X <- X[-1,]
  X[,1] = 1;
  return(X);
}
generateY <- function(X, B, Nk, sigma.2){
  XB <- X%*%B;
  p = length(X[,1]);
  Y <- mvrnorm(1, XB, sigma.2*diag(p));
  return(Y);
}
calcualteS_b <- function(beta0, MCMC.Betas, m, s){
  S_b <- matrix(0, nrow = length(beta0), ncol = length(beta0));
  for(j in 1:m){
    Bj <- MCMC.Betas[[j]][s,];
    S_b <- S_b + (Bj - beta0)%*%t(Bj - beta0);
  }
  return(S_b)
}
updateGamma <- function(dataY, dataX, MCMC.Betas, m , s){
  Build <- 0;
  for(j in 1:m){
    Xj     <- dataX[[j]];
    Yj     <- dataY[[j]];
    Bj <- MCMC.Betas[[j]][s,];
```

```
    Build <- Build + t(Yj - Xj%*%Bj)%*%(Yj - Xj%*%Bj);
  }
  return(Build);
}
```

Generate Data

```
N <- c(1:10)*10; p <- 10; m <- 10;
beta0 <- rep(0, p); sigma.2 <- 1; Sigma0 = diag(p);
dataX <-list();
dataBetas <- matrix(0, nrow = m, ncol = p);
dataY <- list();
for(j in 1:m){
  Xj             <- generateX(N[j],p);
  dataX[[j]]     <- Xj;
  Bj             <- mvrnorm(1, beta0, Sigma0);
  dataBetas[j,] <- Bj;
  Yj <- generateY(Xj, Bj, N[j], sigma.2);
  dataY[[j]] <- Yj;
}
```

Running the Gibbs Sampler

```
Lambda.inv <- diag(p); eta0 <- 2; S0 <- diag(p); nu0 <- 1; sigma0.2 <- 1;
mu <- rep(0, p);
Samples <- 10000;
MCMC.Beta0 <- matrix(0, nrow = Samples, ncol = p);
MCMC.sigma2 <- rep(0, Samples);
MCMC.Sigma0 <- matrix(0, nrow = Samples, ncol = p*p);
MCMC.Betas <- list();
for(j in 1:m){
  MCMC.Betas[[j]]  <- matrix(0, nrow = Samples, ncol = p);
}
#Initial Values
sigma.2 <- 1; beta0 <- rep(0, p); Sigma0 <- diag(p);
#Run the Gibbs
for(s in 1:Samples){
  #Betas 1,..., m
  Sigma0.inv <- solve(Sigma0);
  betaSums <- rep(0, p);
  for(j in 1:m){
    Xj <- dataX[[j]]; Yj <- dataY[[j]];
    Var.j  <- solve(Sigma0.inv + t(Xj)%*%Xj/sigma.2);
    Mean.j <- Var.j%*%(Sigma0.inv%*%beta0 + t(Xj)%*%Yj/sigma.2);
    Beta.j <- mvrnorm(1, Mean.j, Var.j)
    betaSums <- betaSums + Beta.j;
    MCMC.Betas[[j]][s,] <- Beta.j;
  }
  betaAverage <- betaSums/m;
```

```
  #sigma.2
  A <- (nu0 + sum(N))/2;
  B <- (nu0*sigma0.2 + updateGamma(dataY, dataX, MCMC.Betas, m, s))/2;
  sigma.2 <- 1/rgamma(1, A, B);
  MCMC.sigma2[s] <- sigma.2;
  #Beta0
  beta0.var  <- solve(Lambda.inv + m*Sigma0.inv);
  beta0.mean <- beta0.var%*%(Lambda.inv%*%mu+m*Sigma0.inv%*%betaAverage);

  beta0 <- mvrnorm(1, beta0.mean, beta0.var);
  MCMC.Beta0[s,] <- beta0;
  #Sigma0
  Sigma0 <- solve(rWishart(1,eta0+m, solve(S0 + calcualteS_b(beta0, MCMC.Betas,m, s)))[,,1])
  prepSigma0 <- matrix(Sigma0, nrow = 1, ncol = p*p);
  MCMC.Sigma0[s,] <- prepSigma0;
}
```

Part III: Assessing the Convergence

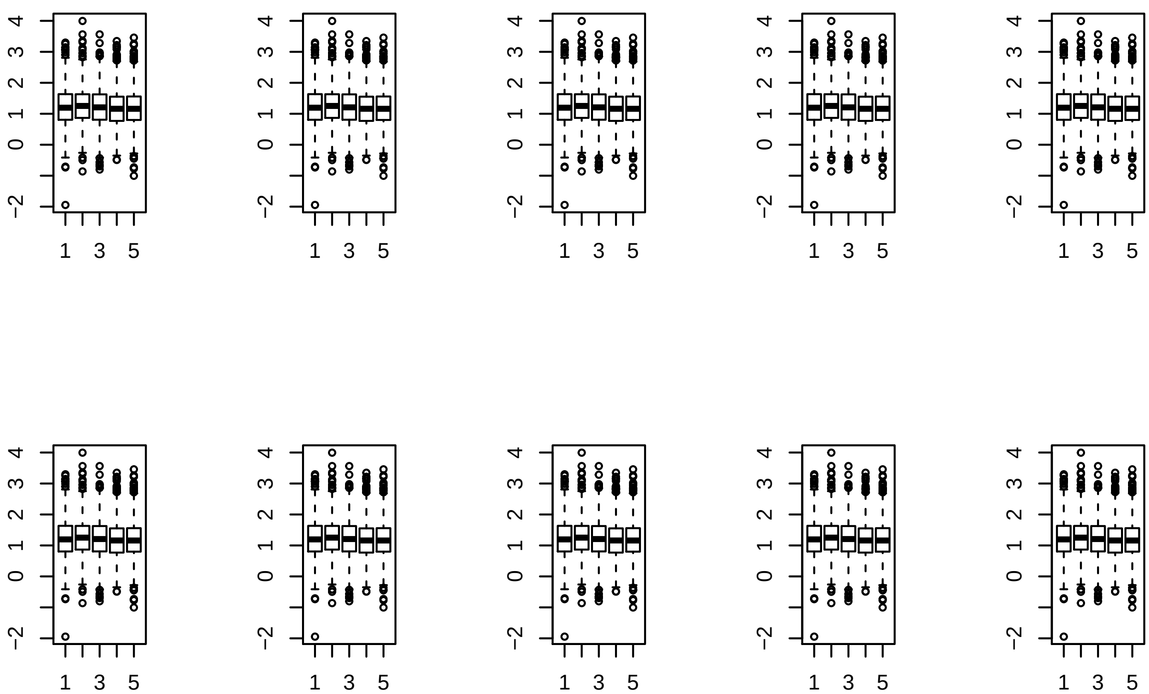
Beta Values of each school

Below, I plot the MCMC average and an MCMC 95% confidence interval for each of the beta values $1, ..., 10$ for each school $1, ..., 10$. I also plot what beta value was acutally used to obtain the data (titled "actual") and the difference between my MCMC expected value and the actual value, and the standard error of each variable as the square root of the variance of the sample divided by the effective sample size. As shown in the boxplots, the beta values seem to have reached a stationary distribution given that the distributions are not experiences large movements or changes over time.

```
library(coda)
betaNames <- c()
for(k in 1:m){
  betaNames <- c(betaNames,paste("Beta", toString(k), sep = ""));
}
for(j in 1:m){
  Summary = matrix(0, nrow = p, ncol = 6)
  colnames(Summary) = c("Actual", "Mean", "2.5%", "97.5%", "Difference %", "Standard Error")
  rownames(Summary) = betaNames
  for(k in 1:p){
    Summary[k, 1]   <- dataBetas[j,k];
    Summary[k, 2]   <- mean(MCMC.Betas[[j]][,k]);
    Summary[k, 3:4] <- quantile(MCMC.Betas[[j]][,k],c(.025, .975));
    Summary[k, 5]   <- (Summary[k, 1] - Summary[k,2])/Summary[k,1]*100;
    Summary[k, 6]   <- sqrt(var(MCMC.Betas[[j]][,k])/effectiveSize(MCMC.Betas[[j]][,k]));
  }
  par(mfrow = c(2,5));
  print(paste("For School ", toString(j)));
  print(Summary)
  print(paste("Boxplots of the convergence for B_", toString(m), ", For each specific Beta"))
  for(k in 1:p){
        boxplot(MCMC.Betas[[j]][,p][1:2000], MCMC.Betas[[j]][,p][2001:4000], MCMC.Betas[[j]][,p][4001:60
```

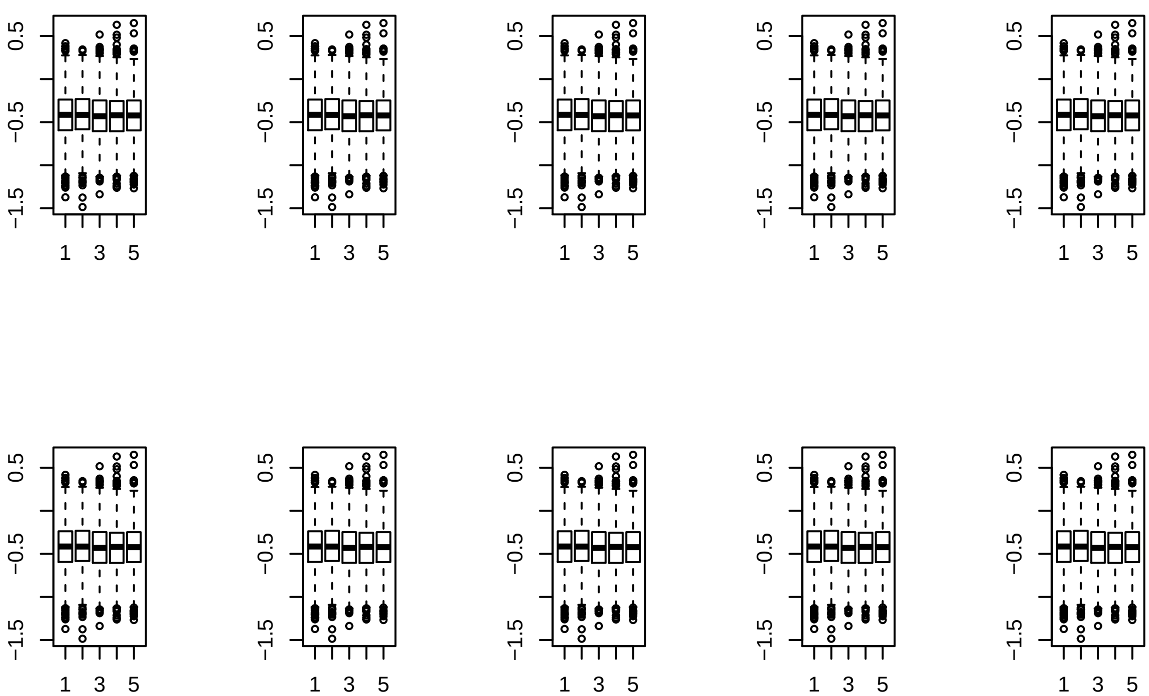
```
        MCMC.Betas[[j]][,p][6001:8000], MCMC.Betas[[j]][,p][8000:10000])
  }
}
```

```
## [1] "For School  1"
##                Actual        Mean       2.5%       97.5% Difference %
## Beta1   0.519232900  0.92731027  0.1748990  1.66800764    -78.59236
## Beta2   0.088852486 -1.59686956 -3.1512436 -0.15910118   1897.21428
## Beta3  -1.126787356 -0.91714508 -1.6881570 -0.17283148     18.60531
## Beta4  -1.427552520  0.39997336 -0.7813500  1.52544890    128.01812
## Beta5  -0.005831412  0.05697641 -1.0643559  1.07009332   1077.06028
## Beta6   0.480801410  0.35432865 -0.5899421  1.33623195     26.30457
## Beta7   1.749163934  1.03853446  0.3358297  1.75599344     40.62681
## Beta8   0.189239838 -1.58087278 -3.3433568  0.09416411    935.38054
## Beta9   0.856832333  0.75826848 -0.1423133  1.76738004     11.50328
## Beta10  0.937914161  1.21588506  0.1116477  2.43496515    -29.63714
##        Standard Error
## Beta1     0.004993955
## Beta2     0.011992020
## Beta3     0.005557258
## Beta4     0.009786467
## Beta5     0.009388476
## Beta6     0.007605691
## Beta7     0.004824568
## Beta8     0.012957533
## Beta9     0.008148100
## Beta10    0.008666417
## [1] "Boxplots of the convergence for B_ 10 , For each specific Beta"
```



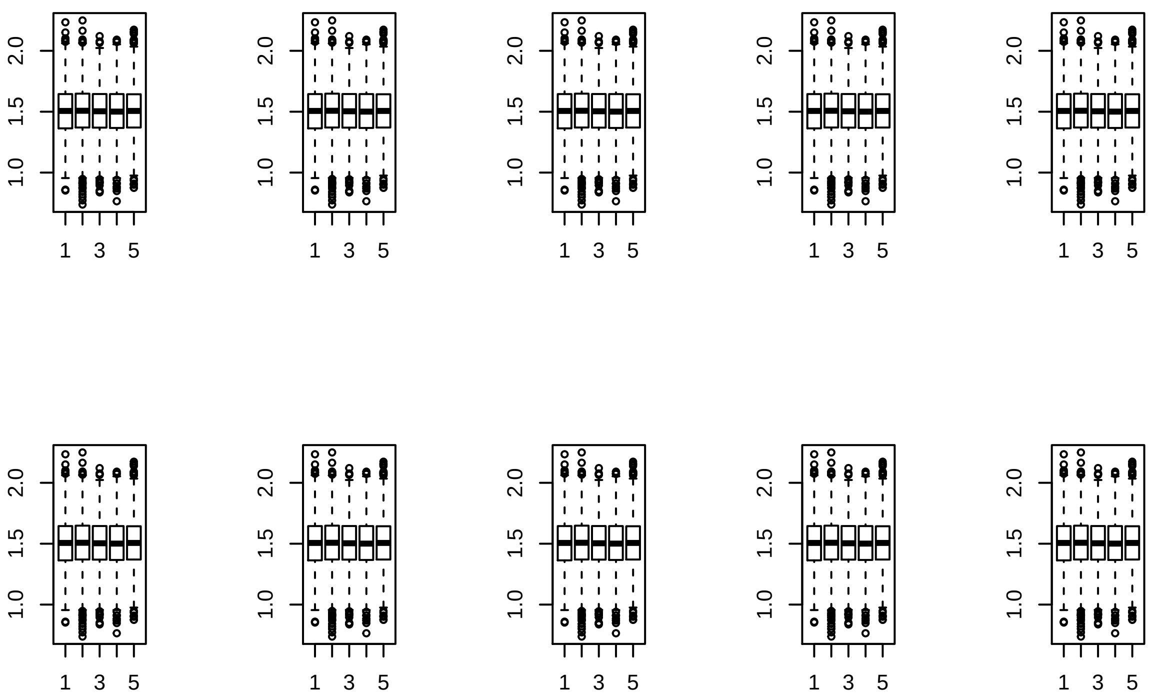
```
## [1] "For School  2"
##              Actual        Mean        2.5%       97.5% Difference %
## Beta1  -0.9348537 -1.13053472 -1.67586067 -0.58505650    -20.93173
```

```
## Beta2  -0.6292481 -0.79869642 -1.53880138 -0.06565357    -26.92870
## Beta3   0.7973415  0.56751447  0.01707696  1.13409088     28.82416
## Beta4  -1.0961474 -0.43818598 -1.15732083  0.27611031     60.02490
## Beta5  -0.1383049  0.07829095 -0.72727338  0.89287320    156.60752
## Beta6   0.5669420  0.71183842  0.05339262  1.37853325    -25.55755
## Beta7  -1.6522887 -1.82479703 -2.79582529 -0.87196355    -10.44057
## Beta8   0.6589718  0.74307257  0.28821794  1.18264636    -12.76242
## Beta9  -0.0861814 -0.10817320 -0.53733910  0.32292986    -25.51803
## Beta10 -0.3548020 -0.42328662 -0.94841650  0.08553807    -19.30221
##        Standard Error
## Beta1     0.003190836
## Beta2     0.005570460
## Beta3     0.003990478
## Beta4     0.005207830
## Beta5     0.006603572
## Beta6     0.004467097
## Beta7     0.007162751
## Beta8     0.002794076
## Beta9     0.002843435
## Beta10    0.003177251
## [1] "Boxplots of the convergence for B_ 10 , For each specific Beta"
```



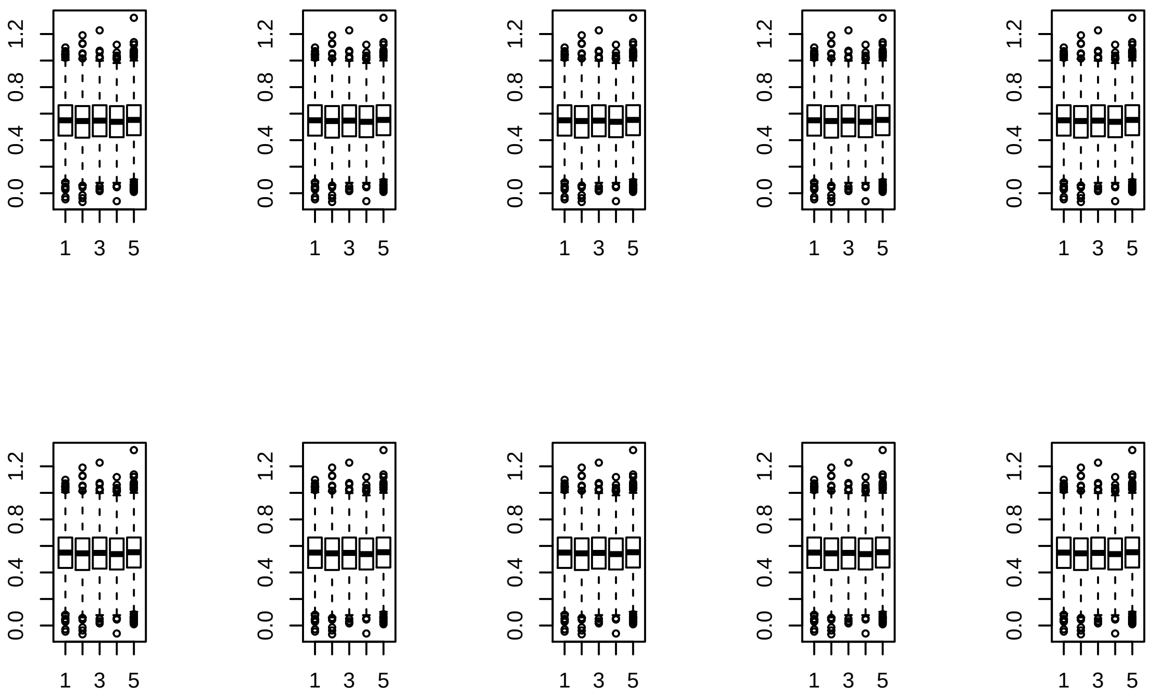
```
## [1] "For School  3"
##            Actual        Mean       2.5%      97.5% Difference %
## Beta1   0.1457298  0.06956212 -0.3558259  0.5079099    52.266378
## Beta2   0.9601534  0.98680820  0.5692211  1.4053055    -2.776099
## Beta3   0.4485389  0.32813607 -0.1843375  0.8315370    26.843339
## Beta4  -0.2379138 -0.48788697 -0.8759205 -0.0980955  -105.068833
## Beta5  -0.5147511 -0.71264750 -1.2958631 -0.1450362   -38.445058
## Beta6   0.4493605  0.71214935  0.3363172  1.0931337   -58.480635
## Beta7   1.1414098  1.31820271  0.8307345  1.8021782   -15.489000
## Beta8  -1.3798826 -1.20795995 -1.5556789 -0.8511520    12.459225
```

```
## Beta9  -1.4105962 -1.92170248 -2.4087193 -1.4388710   -36.233351
## Beta10  1.5377685  1.50608235  1.1015927  1.9126763     2.060528
##        Standard Error
## Beta1     0.002516622
## Beta2     0.002367522
## Beta3     0.002985884
## Beta4     0.002164961
## Beta5     0.003376872
## Beta6     0.002195447
## Beta7     0.002769760
## Beta8     0.001983600
## Beta9     0.002824575
## Beta10    0.002159828
## [1] "Boxplots of the convergence for B_ 10 , For each specific Beta"
```

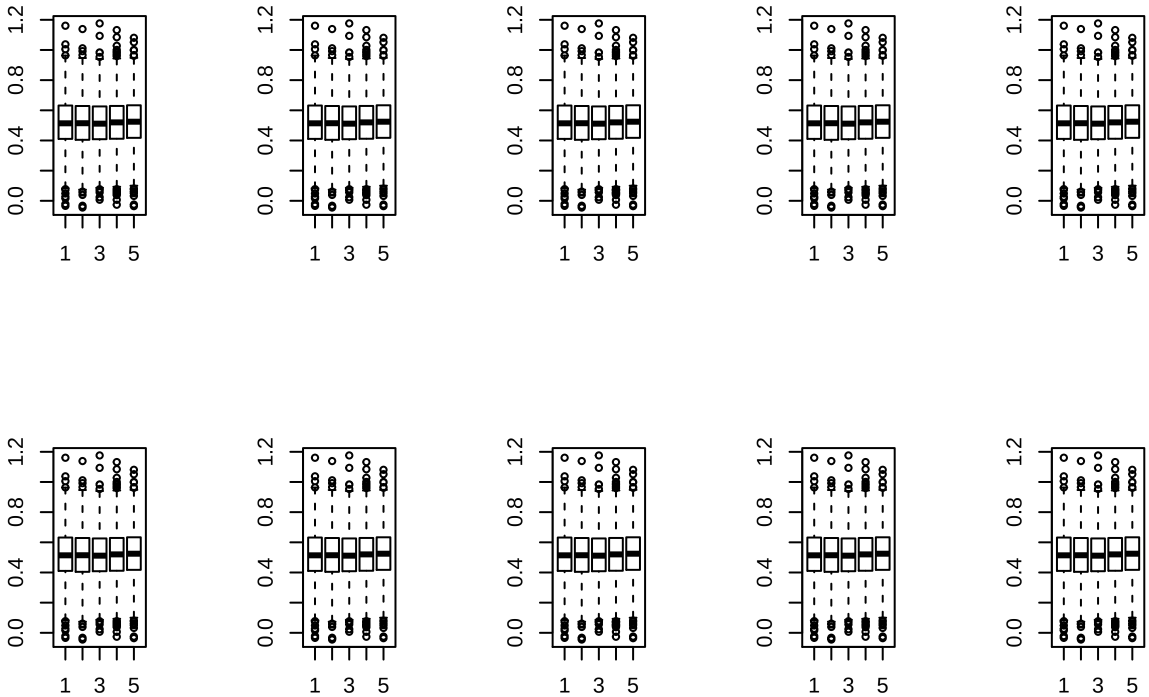


```
## [1] "For School  4"
##              Actual       Mean       2.5%      97.5% Difference %
## Beta1   2.20078106  2.0363121  1.6845794  2.3830286     7.473207
## Beta2  -1.34872828 -1.3925066 -1.7938357 -0.9964340    -3.245894
## Beta3  -0.65423519 -0.7758996 -1.1529016 -0.4050029   -18.596437
## Beta4   0.68051268  0.7519082  0.3801733  1.1202888   -10.491432
## Beta5   1.41392145  1.4283886  0.8439261  2.0135778    -1.023196
## Beta6  -0.05019354 -0.2029185 -0.6015503  0.2055100  -304.272063
## Beta7   1.57945954  1.6424671  1.1337033  2.1536104    -3.989183
## Beta8  -0.71364665 -0.7485914 -1.0841493 -0.4125378    -4.896644
## Beta9   0.65923422  0.9903267  0.6697265  1.3040419   -50.223802
## Beta10  0.67426770  0.5448073  0.1975371  0.8822639    19.200145
##        Standard Error
## Beta1     0.001994208
## Beta2     0.002593176
## Beta3     0.002312574
## Beta4     0.002371436
```

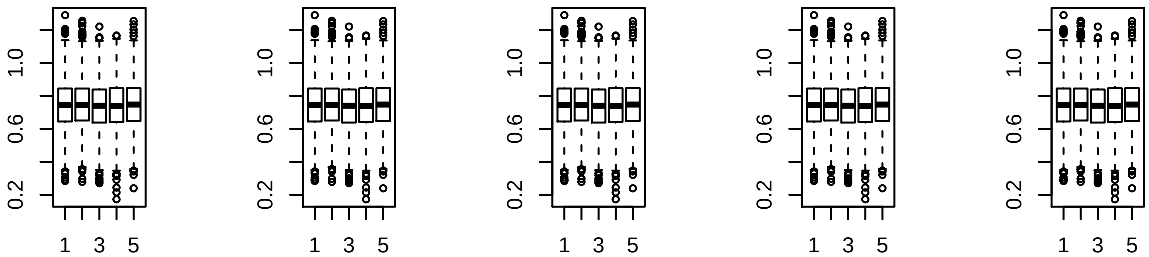
```
## Beta5     0.004228346
## Beta6     0.002434239
## Beta7     0.003623331
## Beta8     0.001837888
## Beta9     0.001922467
## Beta10    0.002085276
## [1] "Boxplots of the convergence for B_ 10 , For each specific Beta"
```

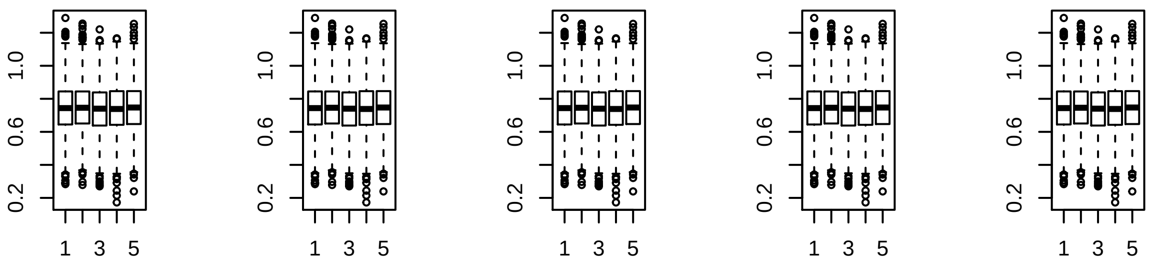



```
## [1] "For School  5"
##             Actual        Mean       2.5%       97.5% Difference %
## Beta1  -2.1075154 -2.17599153 -2.4668178 -1.88325574    -3.249140
## Beta2   1.2629342  1.10798307  0.7880575  1.43359089    12.269138
## Beta3   0.6688175  0.73074705  0.4070425  1.05888971    -9.259555
## Beta4   1.2886718  1.32657195  0.9601763  1.68313648    -2.941025
## Beta5   1.1261540  1.06828296  0.7821985  1.34646401     5.138822
## Beta6   1.5187252  1.53879151  1.2504664  1.82697042    -1.321258
## Beta7   0.3945978  0.53320743  0.2197399  0.84327982   -35.126799
## Beta8  -0.6389493 -0.39370942 -0.7214326 -0.06216811    38.381739
## Beta9  -0.3126930 -0.02973844 -0.3513043  0.29894481    90.489576
## Beta10  0.3663076  0.51921734  0.1999914  0.84071954   -41.743537
##        Standard Error
## Beta1     0.001618933
## Beta2     0.001834312
## Beta3     0.001995988
## Beta4     0.002035877
## Beta5     0.001581895
## Beta6     0.001594858
## Beta7     0.001794524
## Beta8     0.001791070
## Beta9     0.001833565
## Beta10    0.001986575
## [1] "Boxplots of the convergence for B_ 10 , For each specific Beta"
```

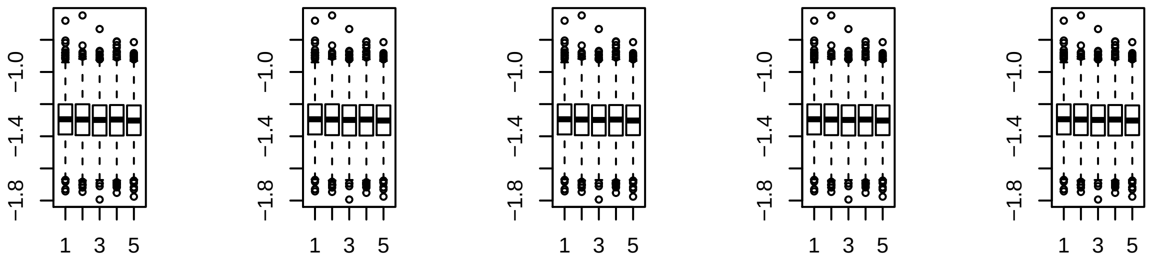


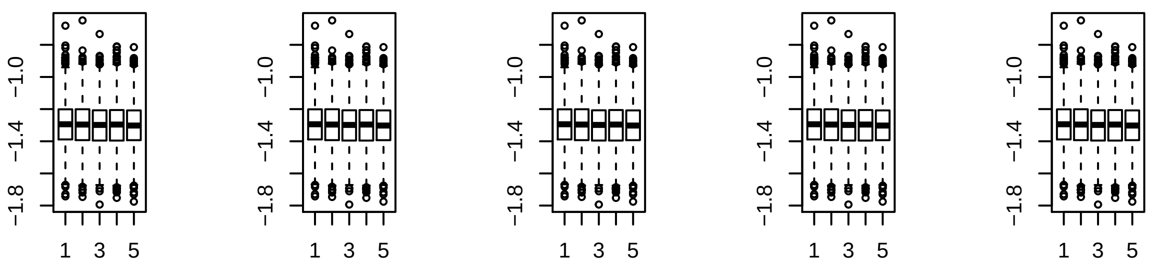

```
## [1] "For School  6"
##             Actual       Mean       2.5%        97.5% Difference %
## Beta1  -0.4321404 -0.2817322 -0.5595491 -0.001358495   34.8054048
## Beta2   0.2953645  0.4749042  0.1957570  0.756605380  -60.7857825
## Beta3   0.5838039  0.5161877  0.2824079  0.752560097   11.5820111
## Beta4  -0.8630841 -0.5961750 -0.8614140 -0.330057289   30.9250359
## Beta5   1.2180149  0.9385340  0.6433931  1.235207874   22.9456014
## Beta6  -0.3273892 -0.3193704 -0.5995298 -0.043225349    2.4493303
## Beta7   1.1433698  1.1525943  0.8604335  1.444620395   -0.8067846
## Beta8   0.3815491  0.4330258  0.1574677  0.715813637  -13.4914854
## Beta9  -1.1144790 -0.9207046 -1.1850086 -0.649081337   17.3869989
## Beta10  0.8292392  0.7448627  0.4586898  1.035083683   10.1751773
##        Standard Error
## Beta1     0.001545908
## Beta2     0.001517713
## Beta3     0.001271350
## Beta4     0.001461964
## Beta5     0.001705992
## Beta6     0.001467553
## Beta7     0.001541088
## Beta8     0.001433680
## Beta9     0.001433258
## Beta10    0.001585299
## [1] "Boxplots of the convergence for B_ 10 , For each specific Beta"
```



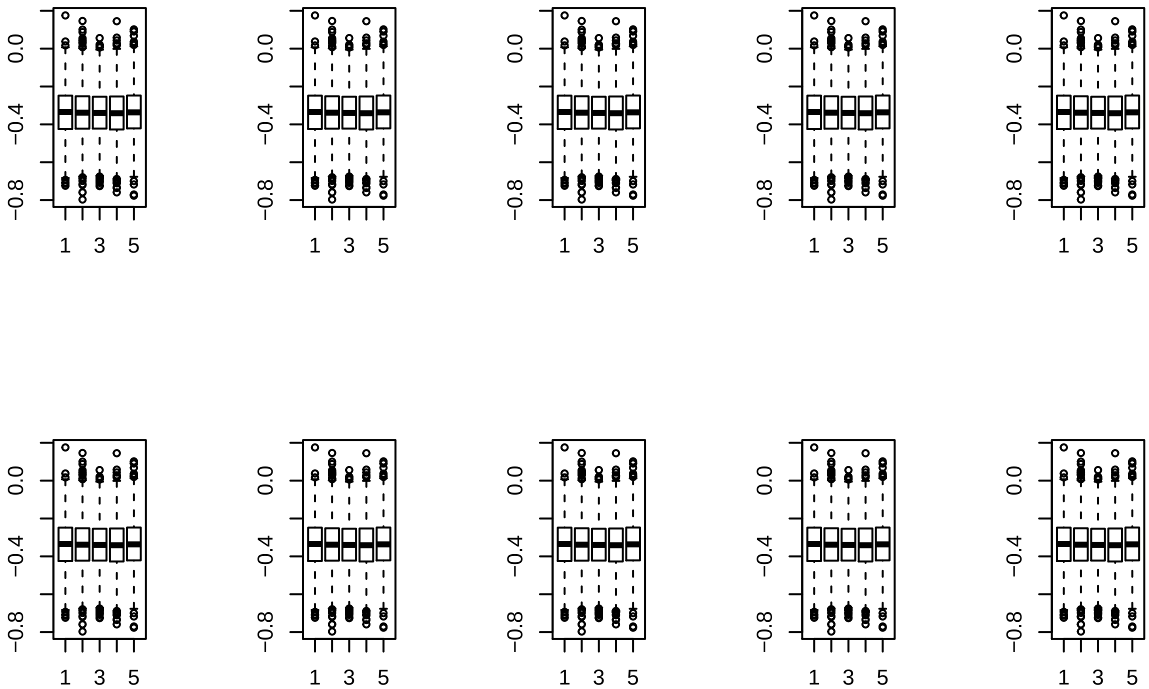


```
## [1] "For School  7"
##             Actual       Mean       2.5%      97.5% Difference %
## Beta1  -0.7423270 -0.8135814 -1.0597069 -0.5620083 -9.598793551
## Beta2   1.7150315  1.5582345  1.2939556  1.8195122  9.142516136
## Beta3  -1.5917220 -1.4412941 -1.6965854 -1.1900649  9.450643639
## Beta4  -0.7642565 -0.8083004 -1.0816178 -0.5378244 -5.762966193
## Beta5  -2.2065053 -2.3026394 -2.5502435 -2.0539116 -4.356850431
## Beta6  -0.7349218 -0.5724435 -0.8579993 -0.2872428 22.108246322
## Beta7   1.0006059  0.8955869  0.5819896  1.2133049 10.495534709
## Beta8   1.0082431  0.9931948  0.7068564  1.2761697  1.492532903
## Beta9  -1.8777272 -1.8776119 -2.1716942 -1.5840906  0.006141473
## Beta10 -1.2321601 -1.2980507 -1.5686687 -1.0222377 -5.347565963
##        Standard Error
## Beta1     0.001287807
## Beta2     0.001477026
## Beta3     0.001363746
## Beta4     0.001490657
## Beta5     0.001417927
## Beta6     0.001515486
## Beta7     0.001797570
## Beta8     0.001510914
## Beta9     0.001627103
## Beta10    0.001483780
## [1] "Boxplots of the convergence for B_ 10 , For each specific Beta"
```

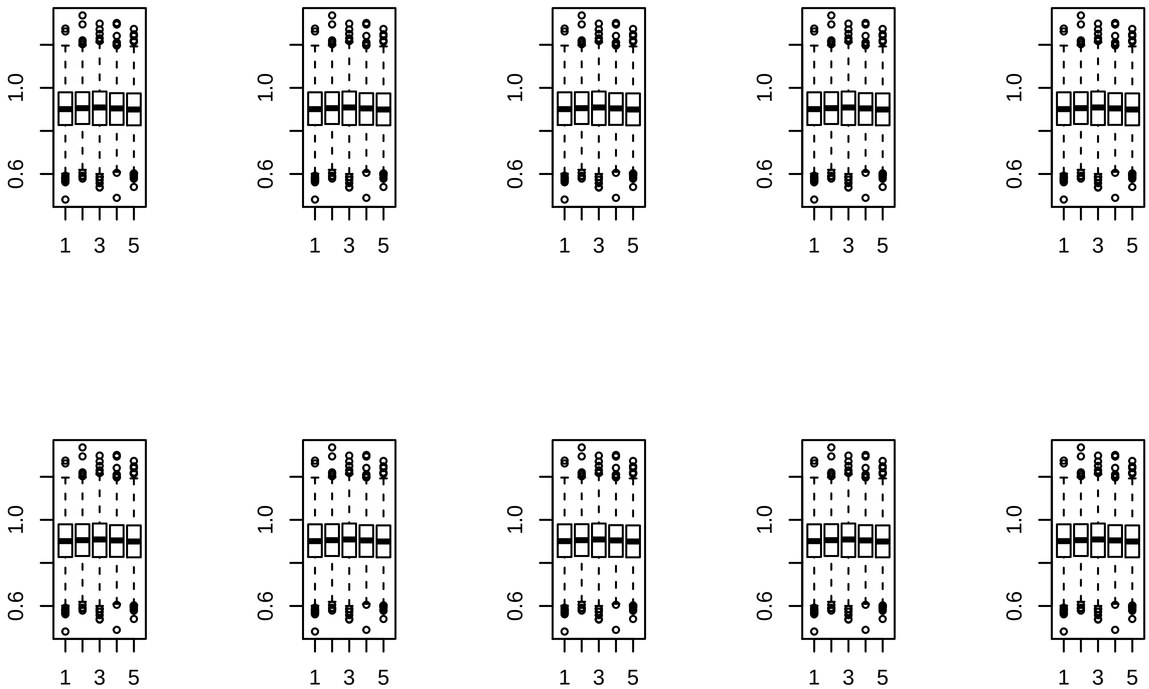




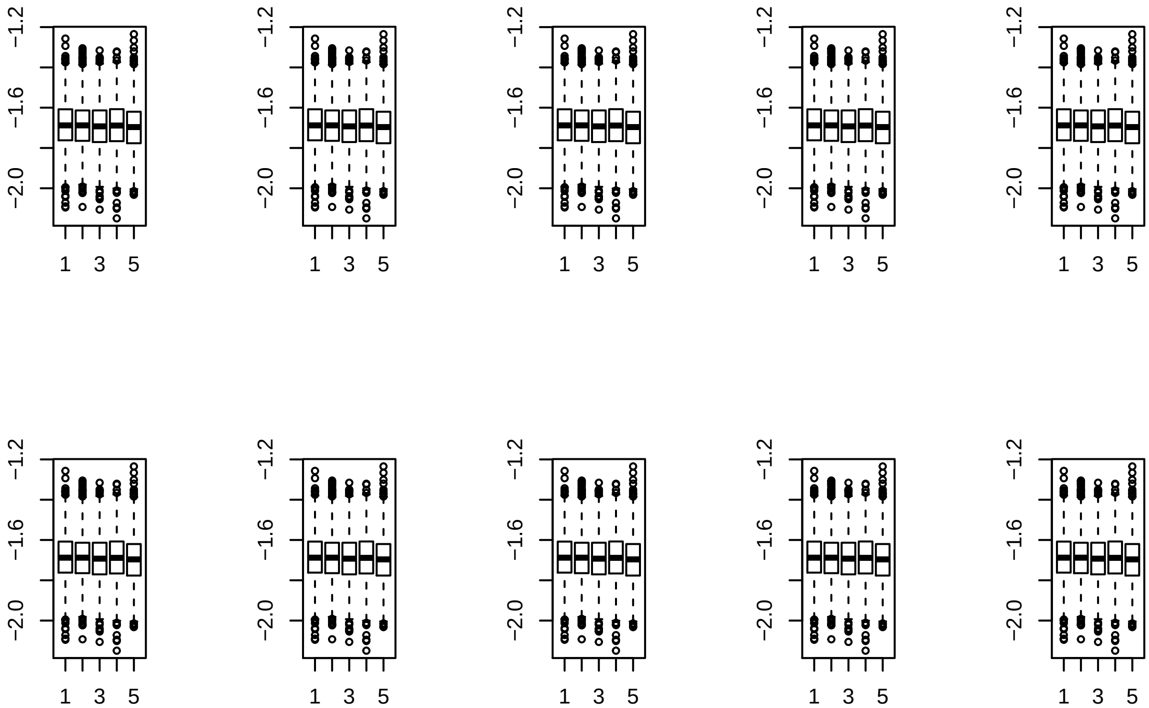
```
## [1] "For School  8"
##             Actual        Mean       2.5%        97.5% Difference %
## Beta1  -0.95737569 -1.11537370 -1.3380809 -0.895153447   -16.503240
## Beta2  -0.23390465 -0.23964805 -0.4792352 -0.004060667    -2.455441
## Beta3   0.17016795 -0.09019615 -0.3479227  0.166140879   153.004196
## Beta4   0.52633096  0.62045360  0.4087780  0.829427531   -17.882787
## Beta5   0.86562423  0.94004577  0.7182160  1.163194304    -8.597441
## Beta6  -0.74917472 -0.77306884 -1.0193983 -0.524140180    -3.189393
## Beta7  -0.04272028 -0.14711651 -0.3917135  0.095228251  -244.371643
## Beta8  -1.20318984 -1.21536039 -1.4729020 -0.955065182    -1.011524
## Beta9   0.82550669  0.64985396  0.4135003  0.891310643    21.278171
## Beta10 -0.24719869 -0.33724668 -0.5904039 -0.085407559   -36.427374
##        Standard Error
## Beta1     0.001161961
## Beta2     0.001259121
## Beta3     0.001442220
## Beta4     0.001094801
## Beta5     0.001172922
## Beta6     0.001377053
## Beta7     0.001292998
## Beta8     0.001347581
## Beta9     0.001323609
## Beta10    0.001359888
## [1] "Boxplots of the convergence for B_ 10 , For each specific Beta"
```



```
## [1] "For School  9"
##             Actual         Mean       2.5%      97.5% Difference %
## Beta1   0.42909843  0.502469732  0.2877525  0.7158266   -17.098944
## Beta2  -1.31081463 -1.380512491 -1.6213831 -1.1374663    -5.317141
## Beta3  -0.47988434 -0.515246433 -0.7176376 -0.3060739    -7.368879
## Beta4   0.08737966 -0.001661287 -0.2542578  0.2459222   101.901229
## Beta5  -0.98818192 -1.238202095 -1.5045531 -0.9754524   -25.301027
## Beta6   0.46749524  0.487206381  0.2646199  0.7033931    -4.216331
## Beta7  -1.24364121 -1.039028439 -1.2689484 -0.8056929    16.452717
## Beta8  -1.67134690 -1.688462280 -1.9047320 -1.4722299    -1.024047
## Beta9  -0.96277190 -0.990078752 -1.2030279 -0.7764755    -2.836274
## Beta10  0.71726403  0.903161707  0.6870032  1.1170965   -25.917607
##        Standard Error
## Beta1     0.001136709
## Beta2     0.001321243
## Beta3     0.001125366
## Beta4     0.001338520
## Beta5     0.001486354
## Beta6     0.001141948
## Beta7     0.001245238
## Beta8     0.001154183
## Beta9     0.001183860
## Beta10    0.001130383
## [1] "Boxplots of the convergence for B_ 10 , For each specific Beta"
```




```
## [1] "For School  10"
##              Actual        Mean        2.5%      97.5% Difference %
## Beta1   0.25985556  0.26687257  0.07325226  0.4643073    -2.700351
## Beta2  -0.21924510  0.04002418 -0.18567538  0.2661480   118.255448
## Beta3  -0.68082173 -0.61479119 -0.88377047 -0.3500978     9.698654
## Beta4   0.90294177  0.97652353  0.74537122  1.2090221    -8.149115
## Beta5   0.17856163  0.23334177  0.03766459  0.4261578   -30.678565
## Beta6  -0.88852376 -1.17141349 -1.36936387 -0.9706997   -31.838173
## Beta7   0.04947061 -0.08070644 -0.28220169  0.1264649   263.140167
## Beta8   1.45787754  1.33862768  1.11125031  1.5595188     8.179690
## Beta9  -0.05710358  0.09953941 -0.15476453  0.3565885   274.313780
## Beta10 -1.45505113 -1.68926522 -1.91315913 -1.4597998   -16.096623
##        Standard Error
## Beta1     0.001037128
## Beta2     0.001213928
## Beta3     0.001496707
## Beta4     0.001236008
## Beta5     0.000991652
## Beta6     0.001057761
## Beta7     0.001121266
## Beta8     0.001187538
## Beta9     0.001380943
## Beta10    0.001233325
## [1] "Boxplots of the convergence for B_ 10 , For each specific Beta"
```



β_0

```r
Summary = matrix(0, nrow = p, ncol = 5)
colnames(Summary) = c("Actual", "Mean", "2.5%", "97.5%", "Standard Error")
rownames(Summary) = betaNames
for(k in 1:p){
  Summary[k, 1]   <- 0;
  Summary[k, 2]   <- mean(MCMC.Beta0[,k]);
  Summary[k, 3:4] <- quantile(MCMC.Beta0[,k],c(.025, .975));
  Summary[k, 5]   <- sqrt(var(MCMC.Beta0[,k])/effectiveSize(MCMC.Beta0[,k]));
}
print("For Beta_0");
```

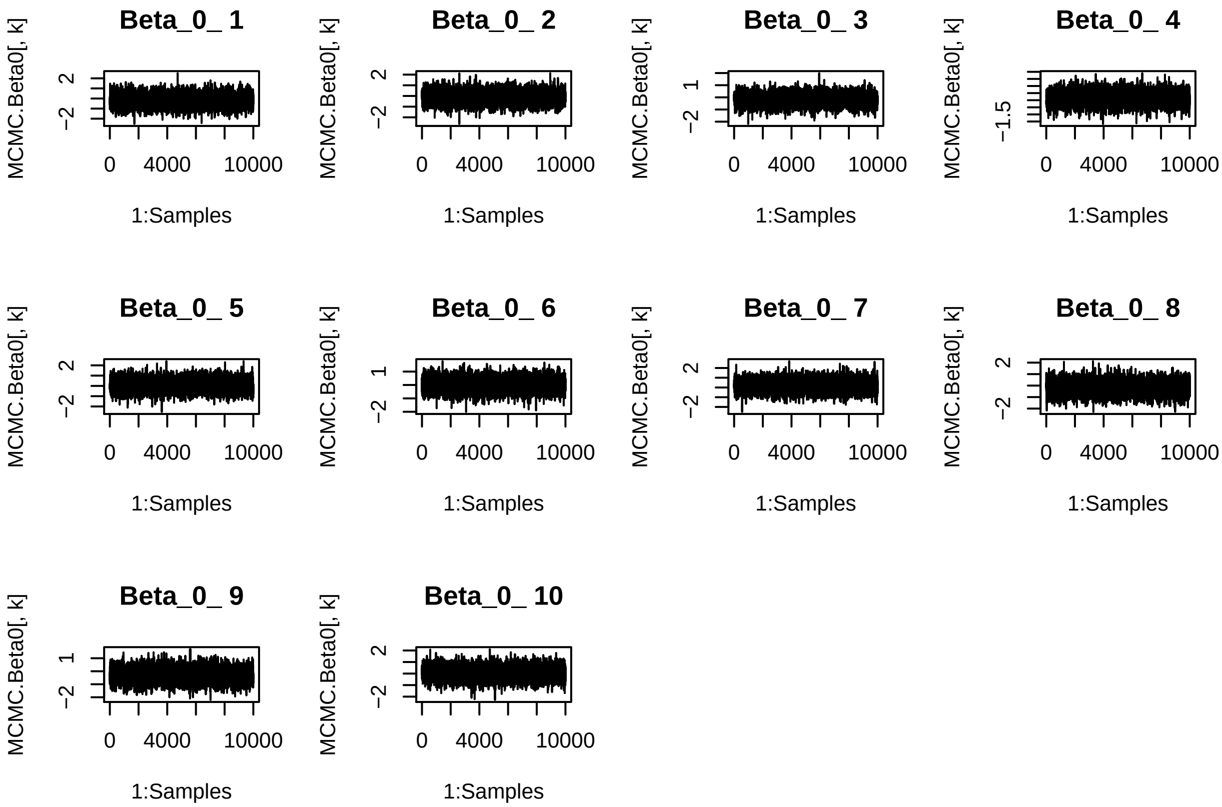
```
## [1] "For Beta_0"
```

```r
print(Summary)
```

```
##        Actual        Mean       2.5%     97.5% Standard Error
## Beta1       0 -0.19096254 -1.2193292 0.8452030    0.005287115
## Beta2       0 -0.12999749 -1.0939260 0.8501786    0.004923669
## Beta3       0 -0.20054704 -0.9596543 0.5592323    0.003972032
## Beta4       0  0.16503459 -0.6123830 0.9208262    0.004135720
## Beta5       0  0.05782741 -0.9199941 1.0879735    0.005506188
## Beta6       0  0.03851145 -0.7960740 0.8519223    0.004313926
## Beta7       0  0.24797811 -0.7855246 1.2548444    0.005378243
## Beta8       0 -0.21675474 -1.1960567 0.7899825    0.005109209
## Beta9       0 -0.29543627 -1.1695508 0.5947194    0.004659723
## Beta10      0  0.06878413 -0.8687981 1.0025311    0.004924480
```

```r
par(mfrow = c(3,4));
for(k in 1:p){
```

```
  plot(1:Samples, MCMC.Beta0[,k], type = 'l', main = paste("Beta_0_", toString(k)))
}
```



σ^2

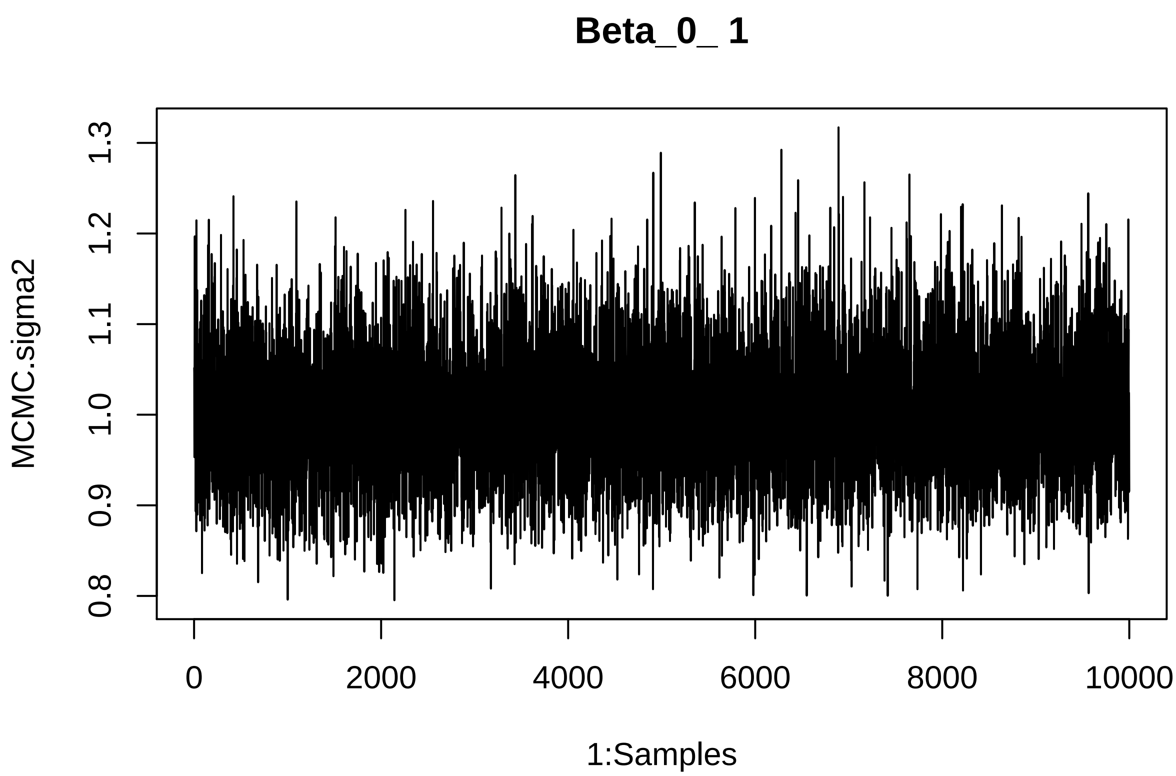
```
Summary = matrix(0, nrow = 1, ncol = 5)
colnames(Summary) = c("Actual", "Mean", "2.5%", "97.5%", "Standard Error")
rownames(Summary) = c("Sigma^2")
for(k in 1:1){
  Summary[k, 1]   <- 1;
  Summary[k, 2]   <- mean(MCMC.sigma2);
  Summary[k, 3:4] <- quantile(MCMC.sigma2,c(.025, .975));
  Summary[k, 5]   <- sqrt(var(MCMC.sigma2)/effectiveSize(MCMC.sigma2));
}
print("For Sigma^2");
```

```
## [1] "For Sigma^2"
```

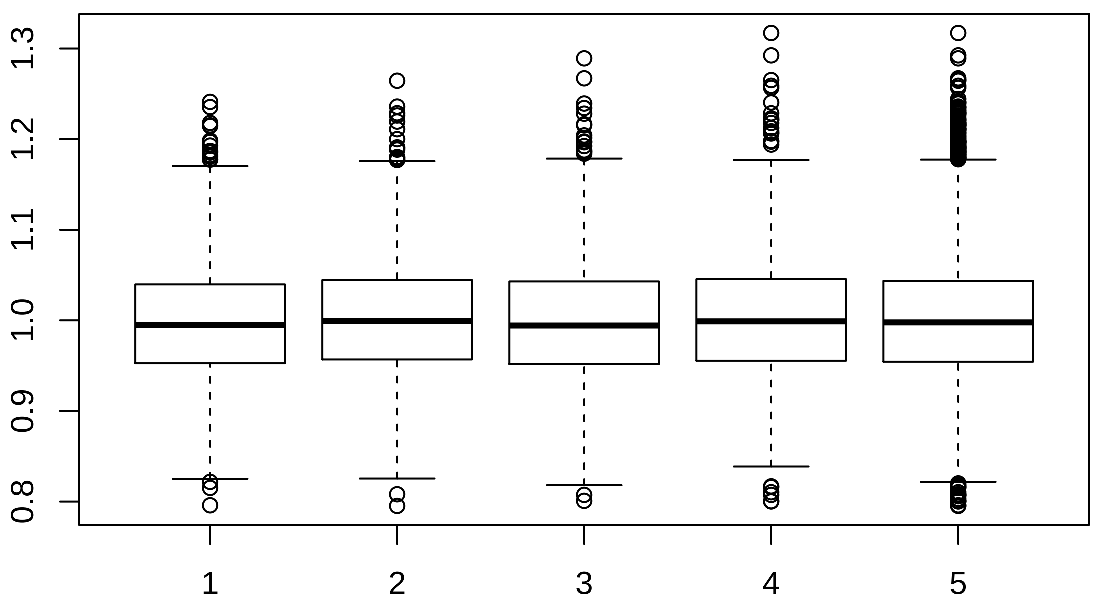
```
print(Summary)
```

```
##           Actual    Mean      2.5%    97.5% Standard Error
## Sigma^2        1 1.00111 0.8788458 1.143155   0.0008024523
```

```
plot(1:Samples, MCMC.sigma2, type = 'l', main = paste("Beta_0_", toString(k)))
```




```r
boxplot(MCMC.sigma2[1:2000], MCMC.sigma2[2001:4000], MCMC.sigma2[4001:6000],
        MCMC.sigma2[6001:8000], MCMC.sigma2[8:10000])
```



The sampler seems to have done an excellent job reaching the stationary distribution of σ^2 as shown in the considency of the distribution of σ^2 values of the boxplots above and the martingale like nature of the movement of the σ^2 value in the traceplot. Also, the standard error is strickingly small and the expected value is close to 1, the acutal value which was used to produce the data.

Σ_0

To assess this, I have converted the p by p covariance matrix into a single vector of length p^2. I will see how the specific values converge and what their distributions look like. The reader may find this large table of 100 row difficult to look at, but allow me to point out two interesting features of the table. First, recall that the acutal Σ_0 that was use to generate the β_j values was an identity matrix. The first thing I would like to point out, is that all of the values which correspond to a location on the diagonal of the true Σ_0 are positive and their confidence intervalse do not contain zero. Secondly, almost all of the off diagonal locations contain zero near the center of their interval. This being said, I feel comfortable with how the Gibbs sampler treated this covariance matrix and am satisified with it's convergence.

```
Summary = matrix(0, nrow = p*p, ncol = 5)
colnames(Summary) = c("Actual", "Mean", "2.5%", "97.5%", "Standard Error")
Covariance <- matrix(.2, nrow = p, ncol = p) + diag(p)*.8
actualSigma0 <- matrix(diag(p), nrow = p*p, ncol = 1, byrow = T);
Summary[,1] <- actualSigma0
for(k in 1:(p*p)){
  Summary[k, 2]   <- mean(MCMC.Sigma0[,k]);
  Summary[k, 3:4] <- quantile(MCMC.Sigma0[,k],c(.025, .975));
  Summary[k, 5]   <- sqrt(var(MCMC.Sigma0[,k])/effectiveSize(MCMC.Sigma0[,k]));
}
print("For Sigma_0");
```

```
## [1] "For Sigma_0"
```

```
print(Summary)
```

```
##        Actual        Mean        2.5%     97.5% Standard Error
##   [1,]      1 16.52219593   1.6654846 82.104413      0.7221185
##   [2,]      0 -8.15899422 -48.4535861  6.269990      0.4251332
##   [3,]      0 -5.11630817 -34.1206328  6.524359      0.4019561
##   [4,]      0  0.26876686 -16.9967281 18.924125      0.2835359
##   [5,]      0  0.79034838 -24.8677386 29.231185      0.3497616
##   [6,]      0 -3.11561483 -28.7217948 11.340000      0.3769940
##   [7,]      0  5.63298119 -13.0314887 39.402125      0.6446985
##   [8,]      0 -3.15538589 -36.3177535 16.520796      0.3744213
##   [9,]      0  3.26380879 -14.6364591 32.410624      0.3528708
##  [10,]      0  3.15857214 -15.7205880 32.602031      0.4909101
##  [11,]      0 -8.15899422 -48.4535861  6.269990      0.4251332
##  [12,]      1 15.06418697   1.5369735 75.666044      0.4934390
##  [13,]      0  1.35932654 -12.6605788 20.918444      0.2809712
##  [14,]      0 -1.98290660 -23.2747109 11.938613      0.2637192
##  [15,]      0 -3.39624646 -36.0519342 16.376525      0.3035081
##  [16,]      0 -0.61267985 -18.6693742 18.788442      0.3095603
##  [17,]      0  3.93267938 -15.6125073 35.899214      0.4129615
##  [18,]      0  6.34133887 -10.7312657 41.439111      0.3872689
##  [19,]      0 -7.66238047 -46.6521304  4.806391      0.3564555
```

```
## [20,]   0 -3.98761163 -32.6490134 14.667043     0.3701713
## [21,]   0 -5.11630817 -34.1206328  6.524359     0.4019561
## [22,]   0  1.35932654 -12.6605788 20.918444     0.2809712
## [23,]   1  7.83162547   0.7080814 37.098766     0.4748745
## [24,]   0 -0.10870467 -12.2202374 12.611852     0.2996679
## [25,]   0  3.74982135  -7.2426108 30.096417     0.3410979
## [26,]   0  3.82916929  -5.5346342 22.466044     0.5041994
## [27,]   0 -2.90950338 -24.2580249 11.516432     0.5453853
## [28,]   0 -0.20935837 -17.2760096 17.672509     0.3244592
## [29,]   0 -0.78670930 -16.2789231 14.136680     0.3435323
## [30,]   0  3.07590741 -10.0051032 22.898525     0.5078646
## [31,]   0  0.26876686 -16.9967281 18.924125     0.2835359
## [32,]   0 -1.98290660 -23.2747109 11.938613     0.2637192
## [33,]   0 -0.10870467 -12.2202374 12.611852     0.2996679
## [34,]   1  7.58876756   0.7409899 34.151831     0.4523006
## [35,]   0  5.20288244  -5.4641030 32.701211     0.3155328
## [36,]   0 -0.28711704 -12.4482407 12.852193     0.4289371
## [37,]   0  0.45121529 -16.7354974 17.468735     0.3202565
## [38,]   0 -1.70617016 -21.7121255 13.402270     0.4641623
## [39,]   0  5.69422602  -3.9408550 32.312112     0.4619285
## [40,]   0 -0.93367893 -17.1789679 14.754457     0.4466434
## [41,]   0  0.79034838 -24.8677386 29.231185     0.3497616
## [42,]   0 -3.39624646 -36.0519342 16.376525     0.3035081
## [43,]   0  3.74982135  -7.2426108 30.096417     0.3410979
## [44,]   0  5.20288244  -5.4641030 32.701211     0.3155328
## [45,]   1 15.10575515   1.5902337 74.908321     0.4590672
## [46,]   0 -0.05212875 -19.4288696 19.958681     0.3184805
## [47,]   0  2.10322764 -20.0037898 31.750048     0.4349129
## [48,]   0 -1.65148561 -29.5762609 22.131518     0.4180054
## [49,]   0  8.47733665  -3.5575341 50.259054     0.3526385
## [50,]   0  2.21325903 -19.2199495 30.991363     0.2663872
## [51,]   0 -3.11561483 -28.7217948 11.340000     0.3769940
## [52,]   0 -0.61267985 -18.6693742 18.788442     0.3095603
## [53,]   0  3.82916929  -5.5346342 22.466044     0.5041994
## [54,]   0 -0.28711704 -12.4482407 12.852193     0.4289371
## [55,]   0 -0.05212875 -19.4288696 19.958681     0.3184805
## [56,]   1  9.18224588   0.8721524 40.153141     0.6840601
## [57,]   0 -1.62642402 -22.4428156 15.762349     0.5485833
## [58,]   0 -4.04726759 -27.6189389 10.427902     0.4717054
## [59,]   0 -1.22793969 -19.0728271 15.322804     0.5041016
## [60,]   0  6.13475651  -5.6265584 32.281229     0.7204304
## [61,]   0  5.63298119 -13.0314887 39.402125     0.6446985
## [62,]   0  3.93267938 -15.6125073 35.899214     0.4129615
## [63,]   0 -2.90950338 -24.2580249 11.516432     0.5453853
## [64,]   0  0.45121529 -16.7354974 17.468735     0.3202565
## [65,]   0  2.10322764 -20.0037898 31.750048     0.4349129
## [66,]   0 -1.62642402 -22.4428156 15.762349     0.5485833
## [67,]   1 15.44391041   1.4395657 75.360364     0.8445346
## [68,]   0 -1.23633304 -28.1729837 20.170012     0.3640382
## [69,]   0 -0.45020242 -22.6026663 22.067768     0.3927891
## [70,]   0  3.33610943 -13.8517047 32.340598     0.5659277
## [71,]   0 -3.15538589 -36.3177535 16.520796     0.3744213
## [72,]   0  6.34133887 -10.7312657 41.439111     0.3872689
## [73,]   0 -0.20935837 -17.2760096 17.672509     0.3244592
```

```
##  [74,]       0 -1.70617016 -21.7121255 13.402270      0.4641623
##  [75,]       0 -1.65148561 -29.5762609 22.131518      0.4180054
##  [76,]       0 -4.04726759 -27.6189389 10.427902      0.4717054
##  [77,]       0 -1.23633304 -28.1729837 20.170012      0.3640382
##  [78,]       1 15.28767746   1.5269473 71.430542      0.7830595
##  [79,]       0 -2.22531395 -27.0944971 17.503326      0.5354935
##  [80,]       0 -9.89546276 -50.8592072  2.669979      0.5666460
##  [81,]       0  3.26380879 -14.6364591 32.410624      0.3528708
##  [82,]       0 -7.66238047 -46.6521304  4.806391      0.3564555
##  [83,]       0 -0.78670930 -16.2789231 14.136680      0.3435323
##  [84,]       0  5.69422602  -3.9408550 32.312112      0.4619285
##  [85,]       0  8.47733665  -3.5575341 50.259054      0.3526385
##  [86,]       0 -1.22793969 -19.0728271 15.322804      0.5041016
##  [87,]       0 -0.45020242 -22.6026663 22.067768      0.3927891
##  [88,]       0 -2.22531395 -27.0944971 17.503326      0.5354935
##  [89,]       1 12.66531632   1.2747366 61.852964      0.5779488
##  [90,]       0 -0.73224465 -21.7574331 19.390544      0.5542875
##  [91,]       0  3.15857214 -15.7205880 32.602031      0.4909101
##  [92,]       0 -3.98761163 -32.6490134 14.667043      0.3701713
##  [93,]       0  3.07590741 -10.0051032 22.898525      0.5078646
##  [94,]       0 -0.93367893 -17.1789679 14.754457      0.4466434
##  [95,]       0  2.21325903 -19.2199495 30.991363      0.2663872
##  [96,]       0  6.13475651  -5.6265584 32.281229      0.7204304
##  [97,]       0  3.33610943 -13.8517047 32.340598      0.5659277
##  [98,]       0 -9.89546276 -50.8592072  2.669979      0.5666460
##  [99,]       0 -0.73224465 -21.7574331 19.390544      0.5542875
## [100,]       1 13.92777606   1.3309641 63.747743      0.8985910
```