STA3105 Bayesian Statistics

Jeong Geonwoo

DUE Friday, December 1

Consider the two candicate models as follows:

$$\mathcal{M}_1 : Y_i = \beta_0 + \beta_1 X_i + \epsilon_i, \quad \epsilon_i \overset{\text{ind}}{\sim} N\left(0, \sigma^2\right)$$
$$\mathcal{M}_2 : Y_i = \beta_0 + \beta_1 X_i^3 + \epsilon_i, \quad \epsilon_i \overset{\text{ind}}{\sim} N\left(0, \sigma^2\right)$$

For both models, we set the priors as

$$\sigma^2 \sim \text{IG}(3, 3), \quad \beta \sim N\left(0, 10^2\right)$$

1. (40 points) In hw6.RData, the simulated response and predictors generated from the true (unknown) model are stored as Y, X, respectively. Using the first 70 observations as a training dataset, implement the MCMC algorithms for $\mathcal{M}_1$ and $\mathcal{M}_2$ separately. Here, you should use adaptMCMC package. For both models, report the followings: - Trace plots, density plots, 95% HPD intervals, posterior mean, acceptance probability, and effective sample size for $\beta_0, \beta_1, \sigma^2$.

First, I loaded dataset and define the first 70 observations as a training dataset.

```r
load("/Users/gunwoojung/Desktop/R_wd/Bayesian_Stat/hw6.RData")
X.train <- X[1:70]; Y.train <- Y[1:70]
X.test <- X[71:100]; Y.test <- Y[71:100]
```

Second, I defined initial value of each parameters $\beta_0, \beta_1, \sigma^2$. And I defined log posterior of each model.

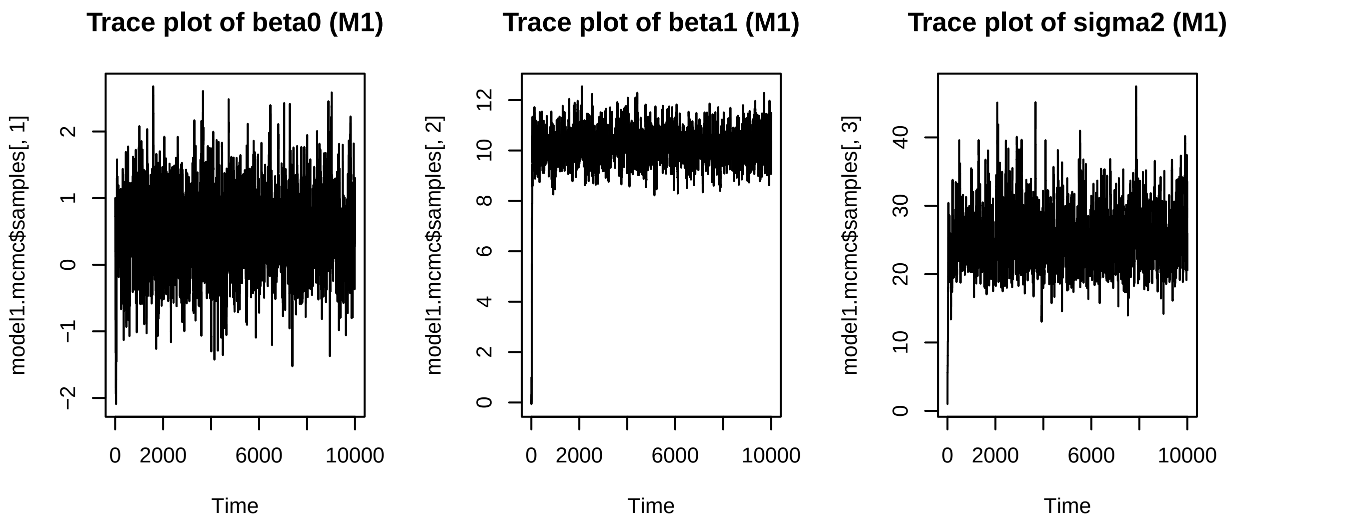
```r
model1.log.posterior <- function(pars){
  with(as.list(pars),{
    beta <- pars[1:2]; sigma2 <- pars[3]
    log.prior.beta <- sum(dnorm(beta, 0, 10, log=TRUE))
    log.prior.sigma2 <- log(dinvgamma(sigma2, 3, 3))
    Y.hat <- beta[1] + beta[2] * X.train
    log.lik <- sum(dnorm(Y.train, mean=Y.hat, sd=sqrt(sigma2), log=TRUE))
    log.posterior <- log.prior.beta + log.prior.sigma2 + log.lik
    return (log.posterior)})}

model2.log.posterior <- function(pars){
  with(as.list(pars),{
    beta <- pars[1:2]; sigma2 <- pars[3]
    log.prior.beta <- sum(dnorm(beta, 0, 10, log=TRUE))
    log.prior.sigma2 <- log(dinvgamma(sigma2, 3, 3))
    Y.hat <- beta[1] + beta[2] * X.train^3
    log.lik <- sum(dnorm(Y.train, mean=Y.hat, sd=sqrt(sigma2), log=TRUE))
    log.posterior <- log.prior.beta + log.prior.sigma2 + log.lik
    return (log.posterior)})}
```

Third, I sampled each parameters $\beta_0, \beta_1, \sigma^2$ using **adaptMCMC** package. I adjusted the acceptance rate = 0.4

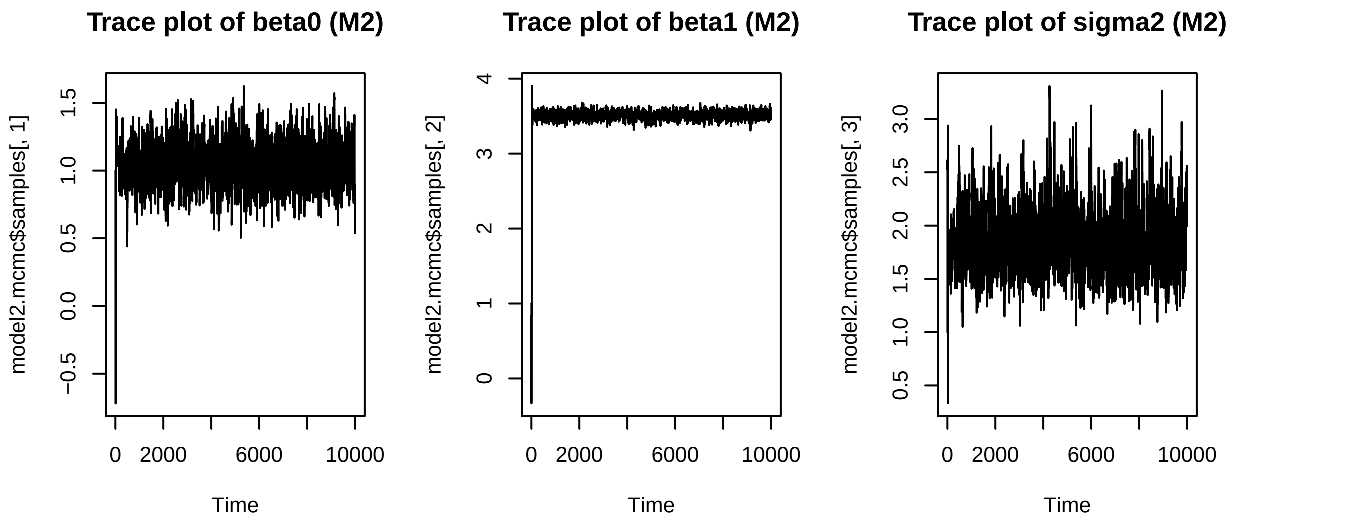
```r
iter <- 10000; init.pars <- c(beta0=1, beta1=1, sigma2=1)
model1.mcmc <- MCMC(p=model1.log.posterior, n=iter, init=init.pars, adapt=TRUE, acc.rate=0.4)
model2.mcmc <- MCMC(p=model2.log.posterior, n=iter, init=init.pars, adapt=TRUE, acc.rate=0.4)
```

Report : Trace plot

```r
# trace plot (for checking burn-in)
par(mfrow=c(1,3))
ts.plot(model1.mcmc$samples[,1], main="Trace plot of beta0 (M1)")
ts.plot(model1.mcmc$samples[,2], main="Trace plot of beta1 (M1)")
ts.plot(model1.mcmc$samples[,3], main="Trace plot of sigma2 (M1)")
```



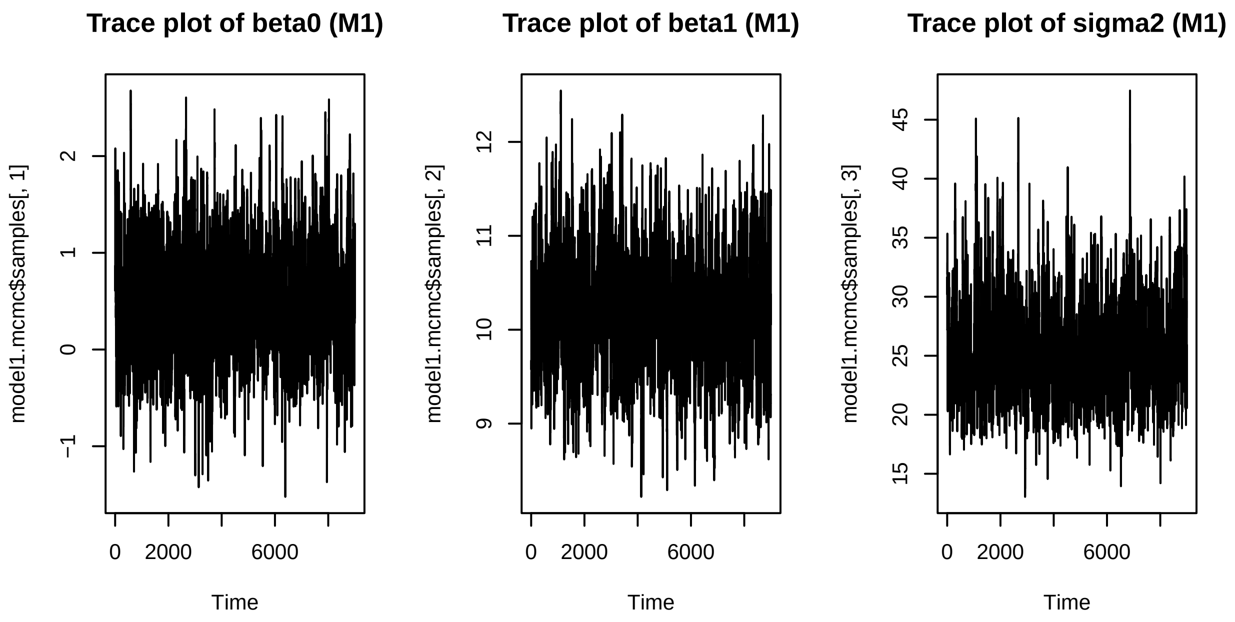
```r
par(mfrow=c(1,3))
ts.plot(model2.mcmc$samples[,1], main="Trace plot of beta0 (M2)")
ts.plot(model2.mcmc$samples[,2], main="Trace plot of beta1 (M2)")
ts.plot(model2.mcmc$samples[,3], main="Trace plot of sigma2 (M2)")
```



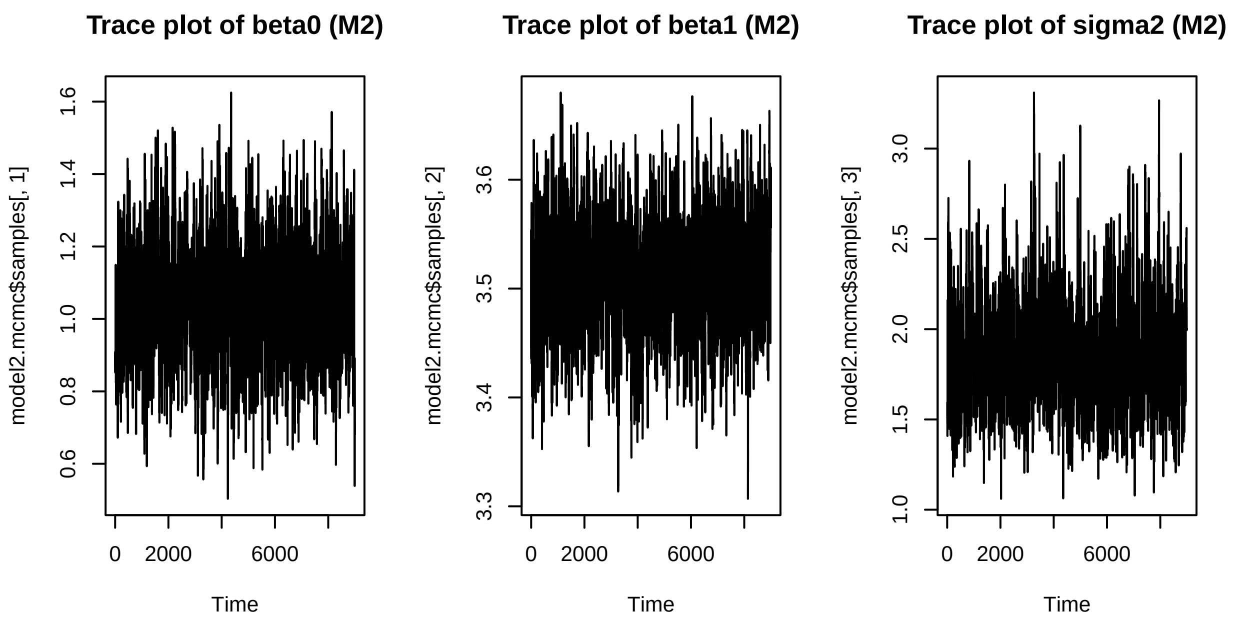
We need to burn-in. I discarded the first 1000 samples of each parameters.

```r
# burn in
burn.in.size = 1000
model1.mcmc$samples <- model1.mcmc$samples[(burn.in.size+1):iter,]
model2.mcmc$samples <- model2.mcmc$samples[(burn.in.size+1):iter,]
```

```r
# trace plot (after burn-in)
par(mfrow=c(1,3))
ts.plot(model1.mcmc$samples[,1], main="Trace plot of beta0 (M1)")
ts.plot(model1.mcmc$samples[,2], main="Trace plot of beta1 (M1)")
ts.plot(model1.mcmc$samples[,3], main="Trace plot of sigma2 (M1)")
```

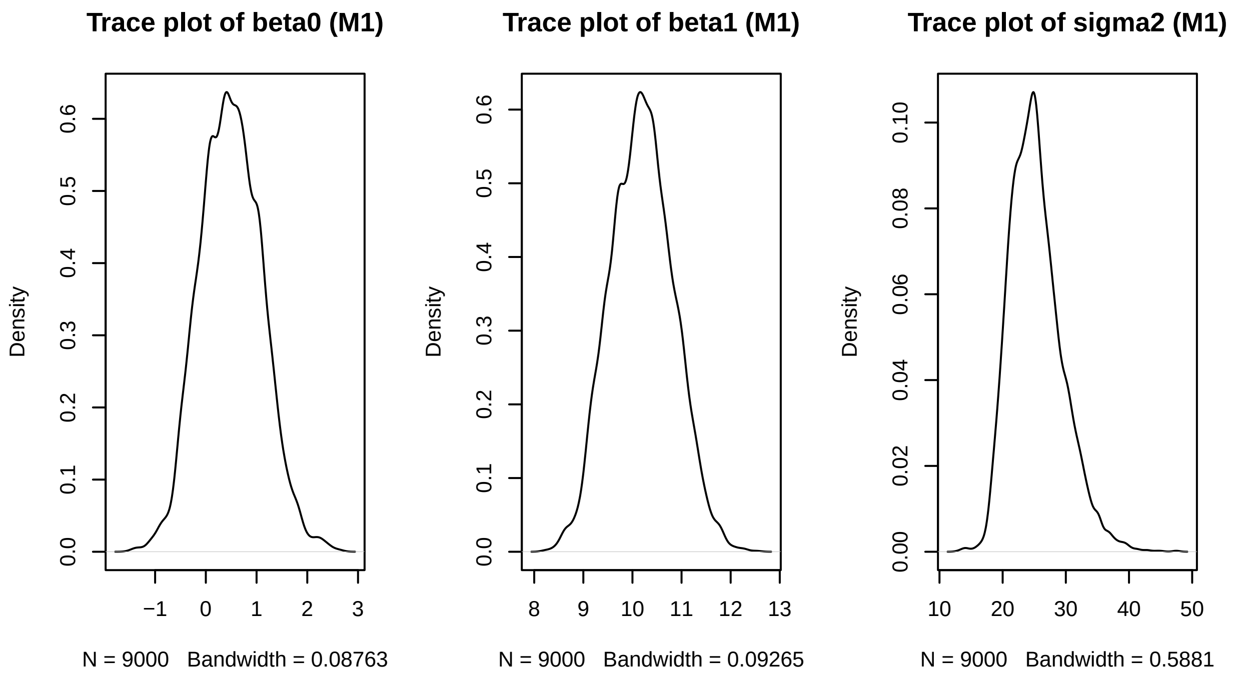


```r
par(mfrow=c(1,3))
ts.plot(model2.mcmc$samples[,1], main="Trace plot of beta0 (M2)")
ts.plot(model2.mcmc$samples[,2], main="Trace plot of beta1 (M2)")
ts.plot(model2.mcmc$samples[,3], main="Trace plot of sigma2 (M2)")
```

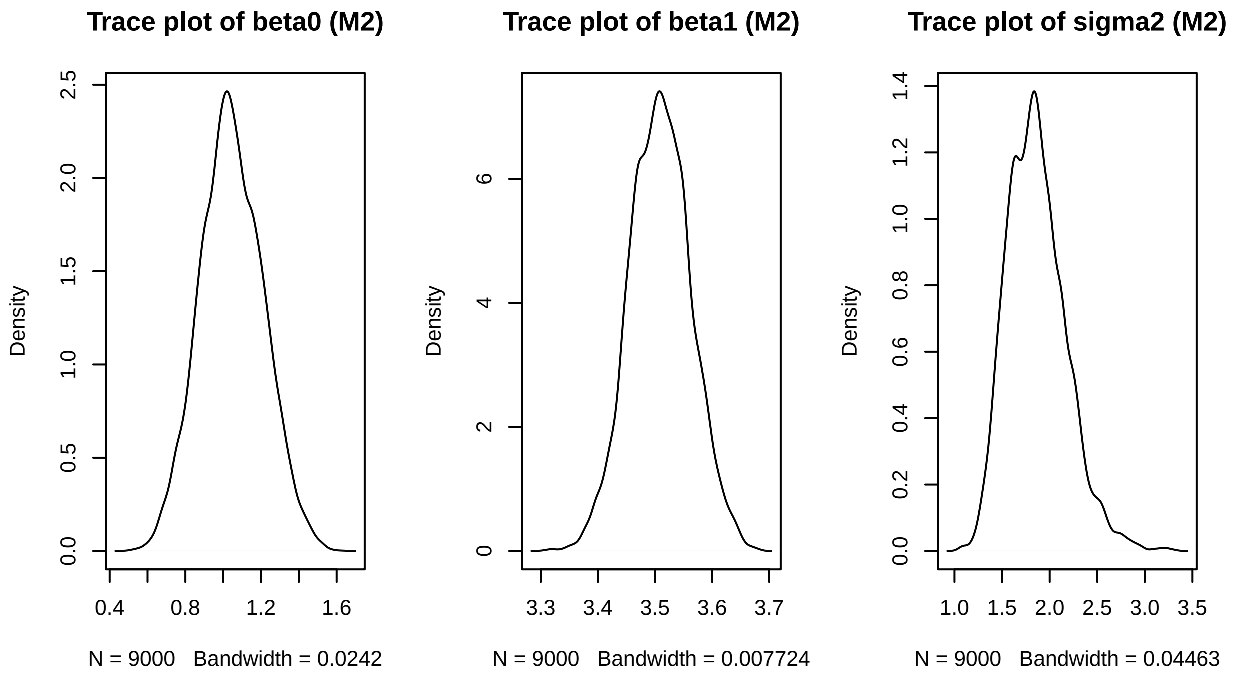


Report : density plot

```
# density plot
par(mfrow=c(1,3))
plot(density(model1.mcmc$samples[,1]), main="Trace plot of beta0 (M1)")
plot(density(model1.mcmc$samples[,2]), main="Trace plot of beta1 (M1)")
plot(density(model1.mcmc$samples[,3]), main="Trace plot of sigma2 (M1)")
```



```
par(mfrow=c(1,3))
plot(density(model2.mcmc$samples[,1]), main="Trace plot of beta0 (M2)")
plot(density(model2.mcmc$samples[,2]), main="Trace plot of beta1 (M2)")
plot(density(model2.mcmc$samples[,3]), main="Trace plot of sigma2 (M2)")
```



Report : 95% HPD intervals

```
cat("95% HPD intervals of beta0 (M1):", HPDinterval(as.mcmc(model1.mcmc$samples[,1]), prob=0.95), "\n",
    "95% HPD intervals of beta1 (M1):", HPDinterval(as.mcmc(model1.mcmc$samples[,2]), prob=0.95), "\n",
    "95% HPD intervals of sigma2 (M1):", HPDinterval(as.mcmc(model1.mcmc$samples[,3]), prob=0.95))
```

```
## 95% HPD intervals of beta0 (M1): -0.5907587 1.675919
##  95% HPD intervals of beta1 (M1): 9.037251 11.44482
##  95% HPD intervals of sigma2 (M1): 17.95719 33.2697
```

```
cat("95% HPD intervals of beta0 (M2):", HPDinterval(as.mcmc(model2.mcmc$samples[,1]), prob=0.95), "\n",
    "95% HPD intervals of beta1 (M2):", HPDinterval(as.mcmc(model2.mcmc$samples[,2]), prob=0.95), "\n",
    "95% HPD intervals of sigma2 (M2):", HPDinterval(as.mcmc(model2.mcmc$samples[,3]), prob=0.95))
```

```
## 95% HPD intervals of beta0 (M2): 0.7354198 1.37498
##  95% HPD intervals of beta1 (M2): 3.406684 3.616192
##  95% HPD intervals of sigma2 (M2): 1.302254 2.463232
```

Report : posterior mean

```
cat("posterior mean of beta0 (M1):", mean(model1.mcmc$samples[,1]), "\n",
    "posterior mean of beta1 (M1):", mean(model1.mcmc$samples[,2]), "\n",
    "posterior mean of sigma2 (M1):", mean(model1.mcmc$samples[,3]))
```

```
## posterior mean of beta0 (M1): 0.4957556
##  posterior mean of beta1 (M1): 10.19351
##  posterior mean of sigma2 (M1): 25.33188
```

```
cat("posterior mean of beta0 (M2):", mean(model2.mcmc$samples[,1]), "\n",
    "posterior mean of beta1 (M2):", mean(model2.mcmc$samples[,2]), "\n",
    "posterior mean of sigma2 (M2):", mean(model2.mcmc$samples[,3]))
```

```
## posterior mean of beta0 (M2): 1.045595
##  posterior mean of beta1 (M2): 3.509295
##  posterior mean of sigma2 (M2): 1.862018
```

Report : acceptance probability

```
cat("acceptance probability of beta0 (M1):", length(unique(model1.mcmc$samples[,1]))/iter, "\n",
    "acceptance probability of beta1 (M1):", length(unique(model1.mcmc$samples[,2]))/iter, "\n",
    "acceptance probability of sigma2 (M1):", length(unique(model1.mcmc$samples[,3]))/iter)
```

```
## acceptance probability of beta0 (M1): 0.356
##  acceptance probability of beta1 (M1): 0.356
##  acceptance probability of sigma2 (M1): 0.356
```

```
cat("acceptance probability of beta0 (M2):", length(unique(model2.mcmc$samples[,1]))/iter, "\n",
    "acceptance probability of beta1 (M2):", length(unique(model2.mcmc$samples[,2]))/iter, "\n",
    "acceptance probability of sigma2 (M2):", length(unique(model2.mcmc$samples[,3]))/iter)
```

```
## acceptance probability of beta0 (M2): 0.364
##  acceptance probability of beta1 (M2): 0.364
##  acceptance probability of sigma2 (M2): 0.364
```

Report : effective sample size

```
cat("effective sample size of beta0 (M1):", effectiveSize(model1.mcmc$samples[,1]), "\n",
    "effective sample size of beta1 (M1):", effectiveSize(model1.mcmc$samples[,2]), "\n",
    "effective sample size of sigma2 (M1):", effectiveSize(model1.mcmc$samples[,3]))
```

```
## effective sample size of beta0 (M1): 758.8776
##  effective sample size of beta1 (M1): 689.5576
##  effective sample size of sigma2 (M1): 674.2257
```

```r
cat("effective sample size of beta0 (M2):", effectiveSize(model2.mcmc$samples[,1]), "\n",
    "effective sample size of beta1 (M2):", effectiveSize(model2.mcmc$samples[,2]), "\n",
    "effective sample size of sigma2 (M2):", effectiveSize(model2.mcmc$samples[,3]))
```

```
## effective sample size of beta0 (M2): 694.1498
##  effective sample size of beta1 (M2): 738.8792
##  effective sample size of sigma2 (M2): 634.1924
```

2. (20 points) Using the results in Problem 1, evaluate Bayesian information criterion (BIC) for $\mathcal{M}_1, \mathcal{M}_2$. Which models are more supported by the dataset?

```r
# Posterior mean for BIC of M1, M2
M1.beta0.mean <- mean(model1.mcmc$samples[,1])
M1.beta1.mean <- mean(model1.mcmc$samples[,2])
M1.sigma2.mean <- mean(model1.mcmc$samples[,3])

M2.beta0.mean <- mean(model2.mcmc$samples[,1])
M2.beta1.mean <- mean(model2.mcmc$samples[,2])
M2.sigma2.mean <- mean(model2.mcmc$samples[,3])

# BIC of M1, M2
M1.Y.hat <- M1.beta0.mean + M1.beta1.mean * X.train
M1.BIC <- -2 * dmvnorm(Y.train, M1.Y.hat, M1.sigma2.mean * diag(length(Y.train)), log = TRUE
                   ) + ncol(model1.mcmc$samples) * log(length(X.train))

M2.Y.hat <- M2.beta0.mean + M2.beta1.mean * X.train^3
M2.BIC <- -2 * dmvnorm(Y.train, M2.Y.hat, M2.sigma2.mean * diag(length(Y.train)), log = TRUE
                   ) + ncol(model2.mcmc$samples) * log(length(X.train))

cat("BIC of M1:", M1.BIC, "||", "BIC of M2:", M2.BIC)
```

```
## BIC of M1: 440.3924 || BIC of M2: 252.9313
```

From the fact that a smaller BIC indicates a better model, we can say that the model 2 is more supported by the dataset than model 1.

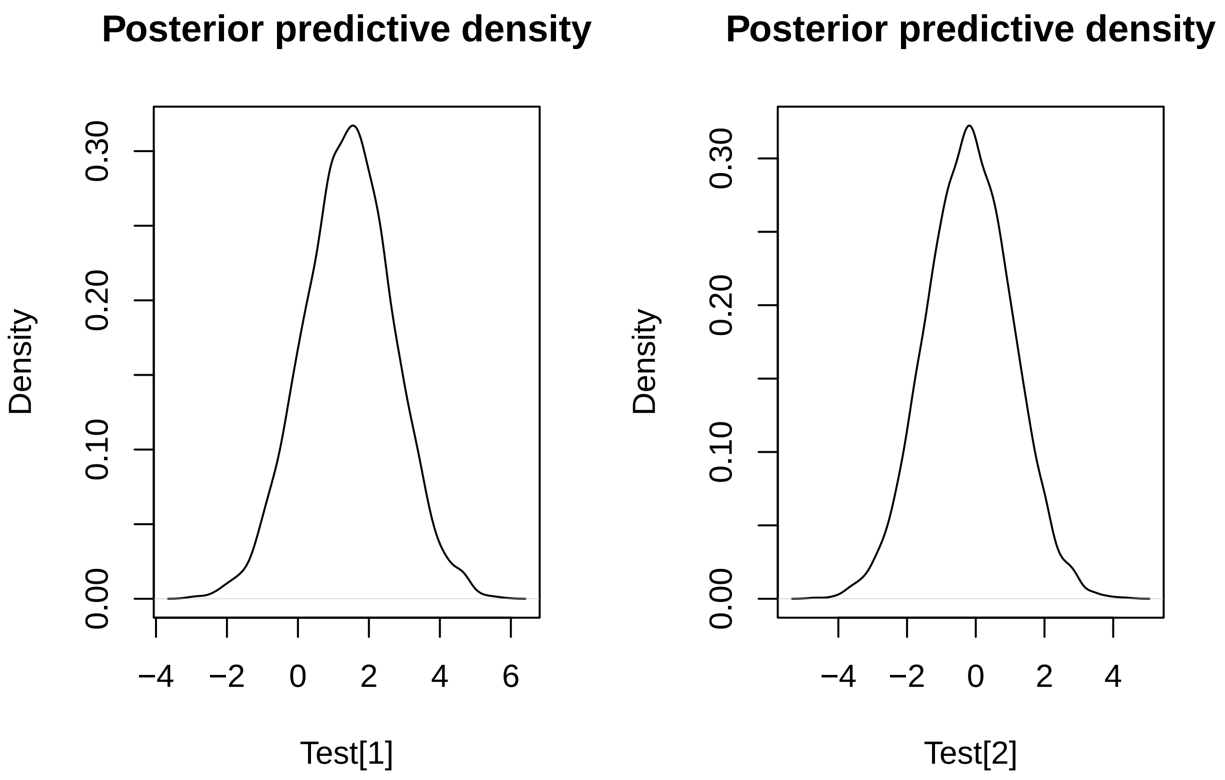
3. (40 points) We will do a prediction based on the remaining 30 observations (i.e., test data). Using the results in Problem 2, predict the response Y (for given X) via Bayesian model averaging (BMA). Report followings:

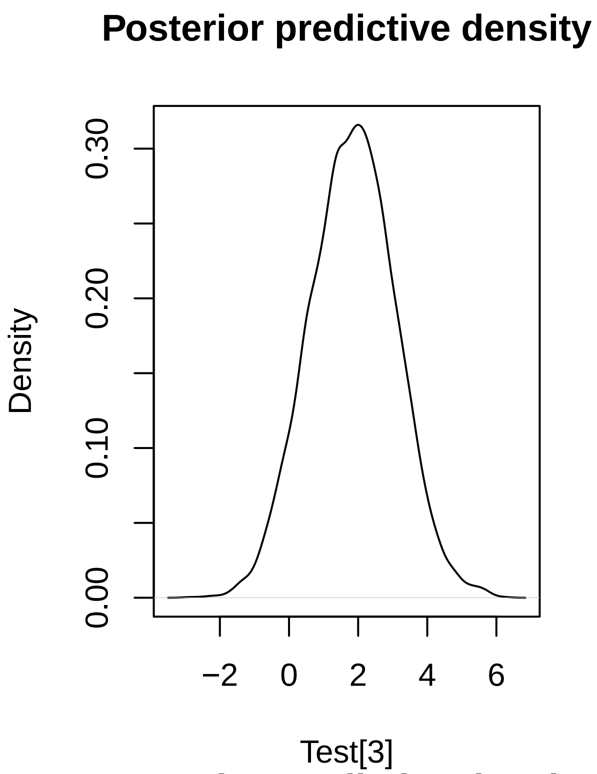
- Draw the density of model-averaged posterior predictive distributions for the first 10 observations in the test dataset.

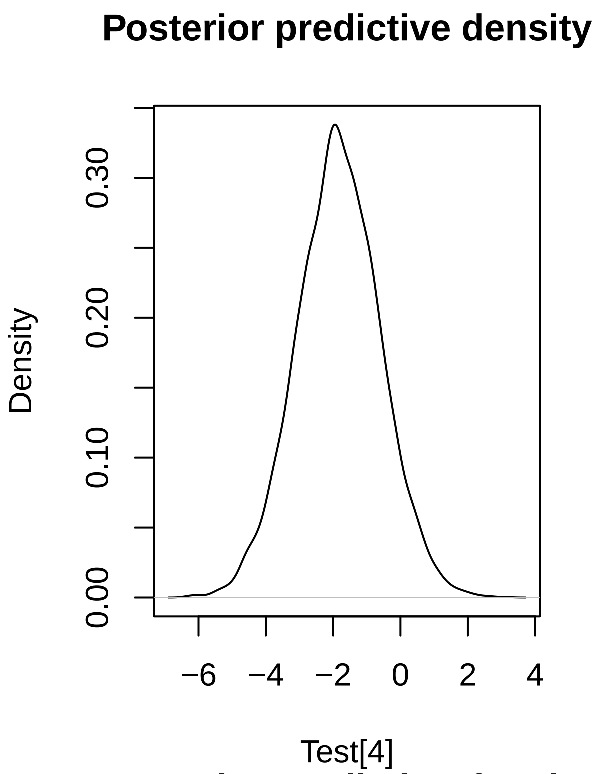
I will report the density of model-averaged posterior predictive distributions (using BIC)

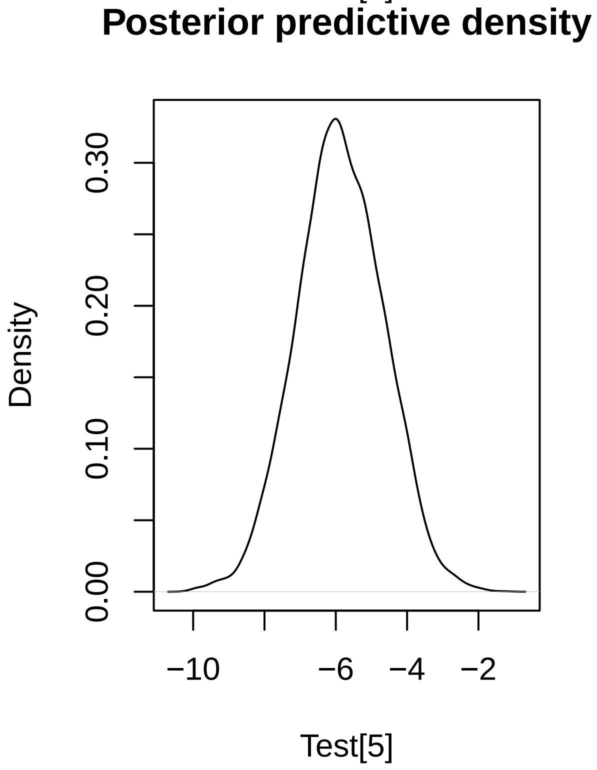
```r
# model prior
M1.prior <- 0.5; M2.prior <- 0.5

# model evidence
M1.evd <- exp(-0.5 * M1.BIC); M2.evd <- exp(-0.5 * M2.BIC)

# Posterior model probability (PMP)
Sum.weight <- M1.evd * M1.prior + M2.evd * M2.prior
M1.weight <- M1.evd * M1.prior / Sum.weight
M2.weight <- M2.evd * M2.prior / Sum.weight
```

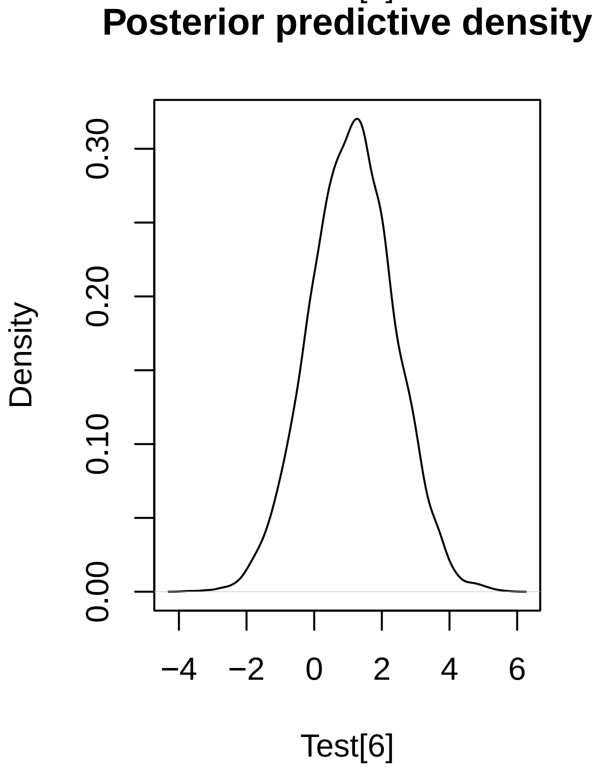
```r
# Posterior predictive distribution Test[1:10]
# I defined a Empty Space "PPD.Test"
# Shape : (iter - burn.in.size) * (length(X.test)) = 9000 * 30
# Ex. Posterior predictive distribution of Test[k] (k=1,...,30)
#     will be saved k-th column of PPT.Test !
PPD.Test <- matrix( rep(NA, (iter - burn.in.size) * length(X.test)) ,
                    ncol=length(X.test))

for (k in 1:length(X.test)){
  M1.PPD.Test.k <- model1.mcmc$samples[,1] + model1.mcmc$samples[,2] * X.test[k] + rnorm(
    9000, 0, sd=sqrt(model1.mcmc$samples[,3][k]))
  M2.PPD.Test.k <- model2.mcmc$samples[,1] + model2.mcmc$samples[,2] * X.test[k]^3 + rnorm(
    9000, 0, sd=sqrt(model2.mcmc$samples[,3][k]))
  PPD.Test[,k] = M1.weight * M1.PPD.Test.k + M2.weight * M2.PPD.Test.k}

for (i in seq(1, 9, 2)) {
  par(mfrow = c(1, 2))
  plot(density(PPD.Test[, i]), main = "Posterior predictive density",
       xlab = paste("Test[", i, "]", sep = ""),  ylab = "Density",)
  plot(density(PPD.Test[, i + 1]), main = "Posterior predictive density",
       xlab = paste("Test[", i + 1, "]", sep = ""), ylab = "Density")}
```

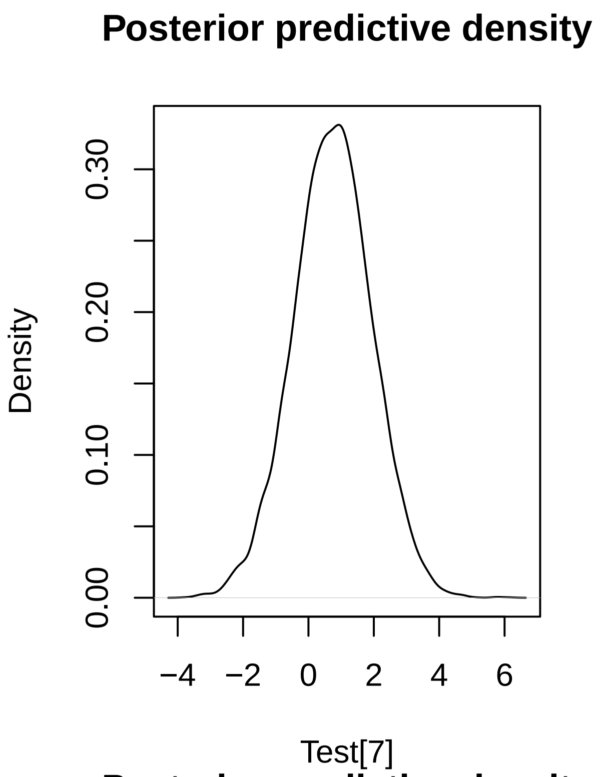


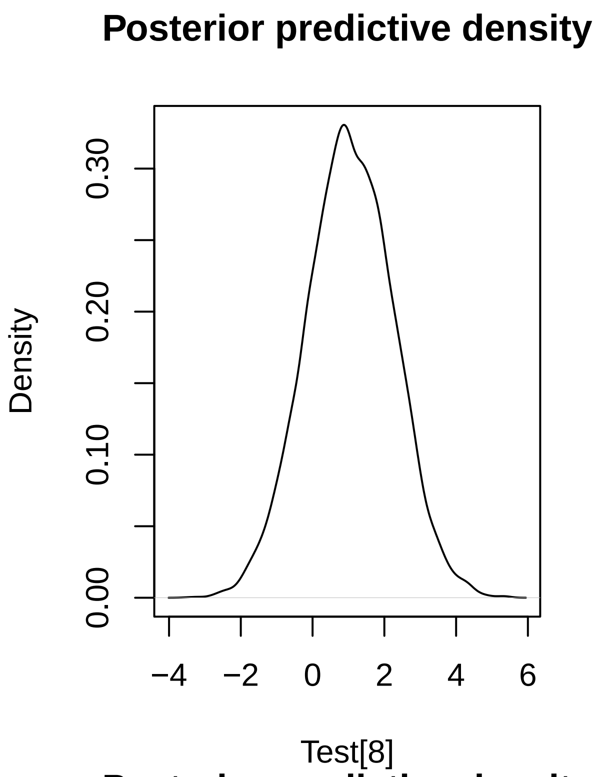

Posterior predictive density
Density
0.00
0.10
0.20
0.30
−2
0
2
4
6
Test[3]

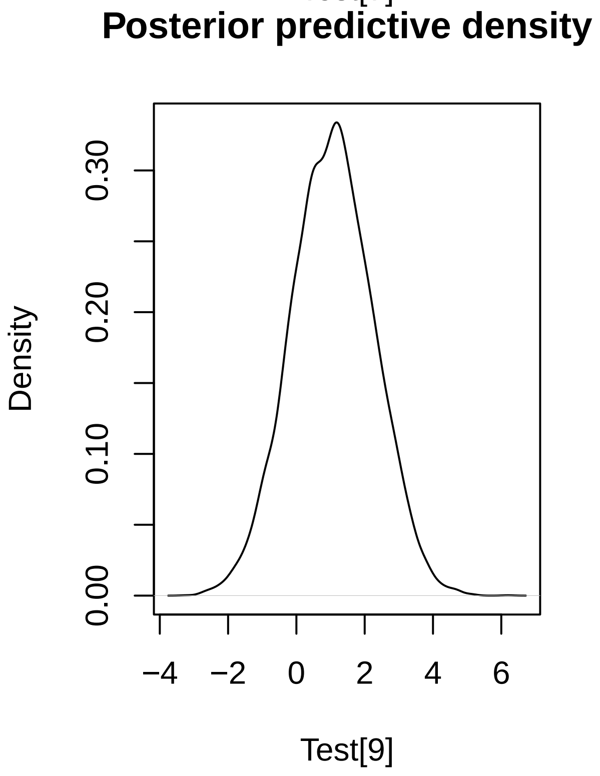

Posterior predictive density
Density
0.00
0.10
0.20
0.30
−6
−4
−2
0
2
4
Test[4]

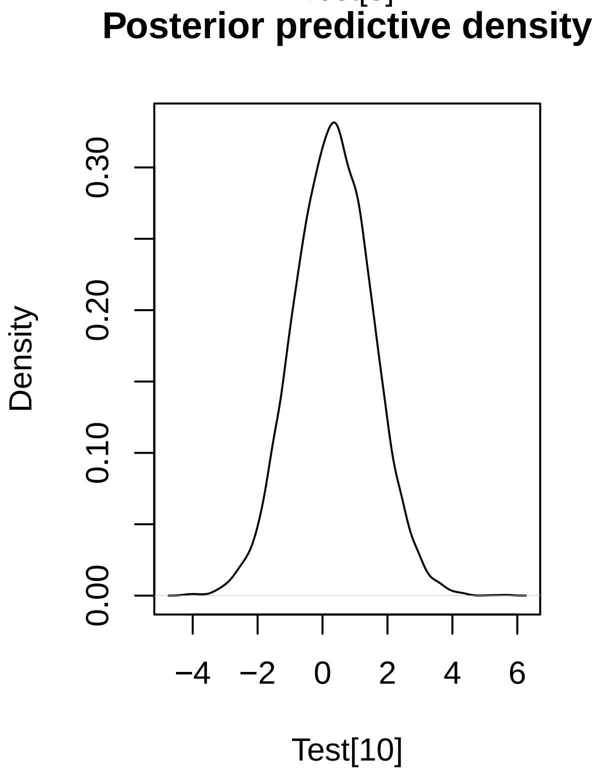

Posterior predictive density
Density
0.00
0.10
0.20
0.30
−10
−6
−4
−2
Test[5]


Posterior predictive density
Density
0.00
0.10
0.20
0.30
−4
−2
0
2
4
6
Test[6]


Posterior predictive density
Density
0.00
0.10
0.20
0.30
−4
−2
0
2
4
6
Test[7]


Posterior predictive density
Density
0.00
0.10
0.20
0.30
−4
−2
0
2
4
6
Test[8]


Posterior predictive density
Density
0.00
0.10
0.20
0.30
−4
−2
0
2
4
6
Test[9]


Posterior predictive density
Density
0.00
0.10
0.20
0.30
−4
−2
0
2
4
6
Test[10]

- Calculate the root mean square prediction error as

$$\sqrt{\sum_{i=1}^{30}\left(Y_{\text{test },i}-Y_{\text{pred },i}\right)^2}$$

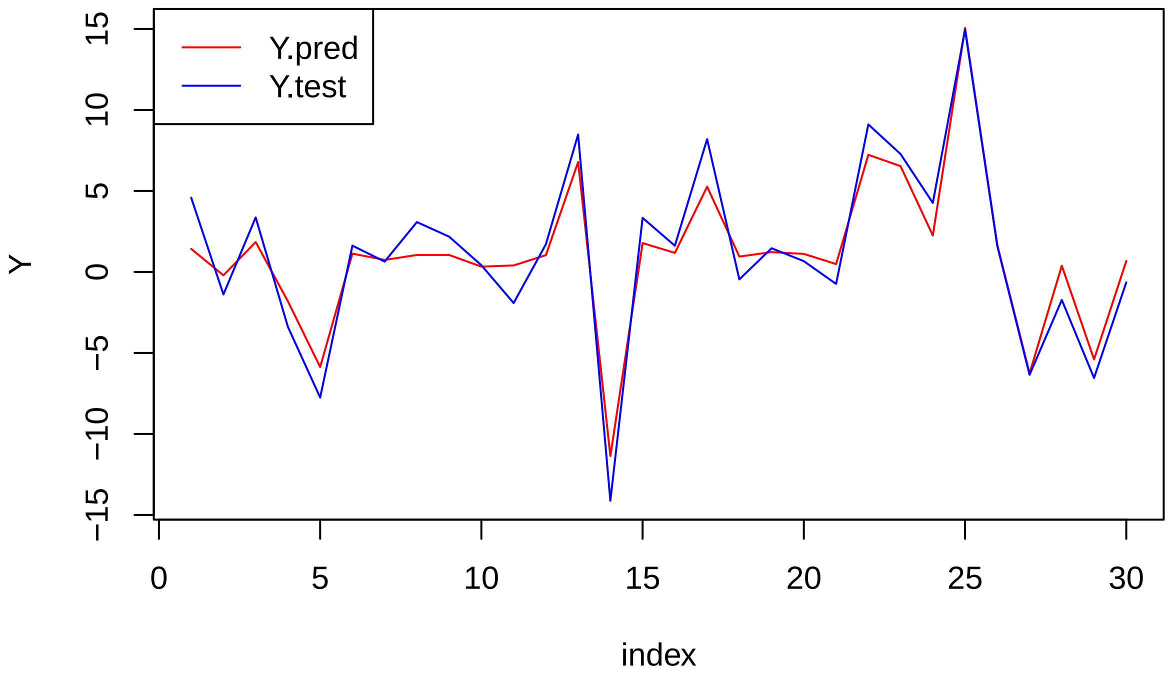
where $Y_{\text{test },i}$ denotes the i th observation of the test dataset and $Y_{\text{pred },i}$ indicates the corresponding model-averaged predicted value from the posterior predictive distribution.

```r
M1.Y.pred <- M1.beta0.mean + M1.beta1.mean * X.test
M2.Y.pred <- M2.beta0.mean + M2.beta1.mean * X.test^3
Y.pred <- M1.weight * M1.Y.pred + M2.weight * M2.Y.pred

cat(sqrt(mean((Y.pred - Y.test)^2)))
```

```
## 1.552479
```

- Draw the plot of the $Y_{\text{test },i}$ versus $Y_{\text{pred },i}$ for $i=1,\cdots,30$.

```r
par(mfrow=c(1,1))
ts.plot(Y.pred, col='red', ylab="Y", xlab="index",
        ylim=c(min(min(Y.pred), min(Y.test)), max(max(Y.pred), max(Y.test))))
lines(Y.test, col='blue')
legend("topleft", legend = c("Y.pred", "Y.test"),
       col = c("red", "blue"), lty = 1, cex = 1)
```


Y.pred
Y.test
Y
index