Econometria Baysiana - Take Home Exam

13/02/2020

Question

Consider the regression model $y_i = x_i'\beta + \epsilon_i 0 \; \epsilon_i|\lambda_i, \sigma^2 \sim N(0, \lambda_i \sigma^2)$ where $x_i = (1, x_{i1}, \ldots, x_{iq})'$ is a p-dimensional vector of regressors (constant plus q attributes or characteristics) and the following hierarchical prior for the scale-mixing variables λ_i:

$$\lambda_1, \ldots, \lambda_n \sim \text{iid Exponential}(1/2)$$

PART A)

We will show that

$$p\left(\epsilon_i|\sigma^2\right) = \int_0^\infty p\left(\epsilon_i|\lambda_i, \sigma^2\right) p\left(\lambda_i\right) d\lambda_i = \frac{1}{2\sigma}\exp\left\{-\frac{|\epsilon_i|}{\sigma}\right\}$$

First consider $p\left(\epsilon_i|\sigma^2\right)$, then we must have that:

$$\begin{aligned}
p\left(\epsilon_i|\sigma^2\right) &= \int_0^\infty p\left(\epsilon_i|\lambda_i, \sigma^2\right) p\left(\lambda_i\right) d\lambda_i \\
&= \int_0^\infty \left(2\pi\lambda_i\sigma^2\right)^{-1/2} \exp\left[-\epsilon_i^2/\left(2\lambda_i\sigma^2\right)\right](1/2)\exp(-\lambda_i/2)d\lambda_i \\
&= (1/2)\left(2\pi\sigma^2\right)^{-1/2} \int_0^\infty \lambda_i^{-1/2} \exp\left[-(1/2)\left(\lambda_i + [\epsilon_i/\sigma]^2\lambda_i^{-1}\right)\right]d\lambda_i
\end{aligned}$$

Now, make a change of variable and let $\psi_i = \lambda_i^{1/2}$. We can then express this integral as

$$p\left(\epsilon_i|\sigma^2\right) = \left(2\pi\sigma^2\right)^{-1/2} \int_0^\infty \exp\left(-[1/2]\left(\psi_i^2 + [\epsilon_i/\sigma]^2\psi_i^{-2}\right)\right)d\psi_i$$

The integral in above can be evaluated analytically. Using the following result from Andrews and Mallows (1974)

$$\int_0^\infty \exp\left\{-0.5\left(a^2u^2 + b^2u^{-2}\right)\right\}du = \left(\frac{\pi}{2a^2}\right)^{1/2} \exp\{-|ab|\}$$

Then $a = 1$, $b = \epsilon_i/\sigma$, and $u = \psi_i$

$$p\left(\epsilon_i|\sigma^2\right) = \left(2\pi\sigma^2\right)^{-1/2}\left(\frac{\pi}{2}\right)^{1/2} \exp\{-|\epsilon_i/\sigma|\} = \frac{1}{2\sigma}\exp\left\{-\frac{|\epsilon_i|}{\sigma}\right\}$$

PART B)

Let $y = (y_1, \dots, y_n)'$ and $X = (x_1, \dots, x_n)'$. Where we have the the following independent priors for $\beta \sim N(\beta_0, V_0)$ and $\sigma^2 \sim IG\left(\frac{\nu_0}{2}, \frac{\nu_0\sigma_0^2}{2}\right)$. Also let $\mathcal{D} = \{y; X\}$, and Λ be denoted as follows

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \lambda_n \end{bmatrix}$$

To implement the Gibbs sampler we need to obtain the complete posterior conditionals for the parameters β, σ^2, and $\{\lambda_i\}_{i=1}^n$ and cycle through the posteriors conditional distributions. The joint posterior distribution is given as

$$p\left(\beta, \{\lambda_i\}, \sigma^2 | y\right) \propto \left[\prod_{i=1}^n \phi\left(y_i; x_i\beta, \lambda_i\sigma^2\right) p\left(\lambda_i\right)\right] p(\beta) p\left(\sigma^2\right)$$

We know that traditional GLS have that $\beta = (X^T\Lambda^{-1}X)^{-1}X^T\Lambda^{-1}y$. If $\beta \sim N(\beta_0, V_0)$, and $\sigma^2 \sim IG\left(\frac{\nu_0}{2}, \frac{\nu_0\sigma_0^2}{2}\right)$ then from the joint posterior, the following complete conditional posterior distributions are obtained:

The β conditional distribution ($\beta|\{\lambda_i\}, \sigma^2, \mathcal{D}$)

$$\begin{aligned} p\left(y|X, \beta, \sigma^2, \{\lambda_i\}\right) &\propto \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^n \hat{\epsilon}_i^2\right\} \\ &\propto \exp\left\{-\frac{1}{2\sigma^2}\left[y^Ty - 2\beta^TX^T\Lambda^{-1}y + \beta^TX^T\Lambda^{-1}X\beta\right]\right\} \\ p\left(\beta|\{\lambda_i\}, \sigma^2, \mathcal{D}\right) &\propto p\left(y|X, \beta, \sigma^2\right) \times p(\beta) \\ &\propto \exp\left\{-\frac{1}{2\sigma^2}\left(-2\beta^TX^T\Lambda^{-1}y + \beta^TX^T\Lambda^{-1}X\beta\right) - \frac{1}{2}\left(-2\beta^TV_0^{-1}\beta_0 + \beta^TV_0^{-1}\beta\right)\right\} \\ &\propto \exp\left\{-\frac{1}{2}\left(-2\beta^T(X^T\Lambda^{-1}y/\sigma^2 + V_0^{-1}\beta_0) + \beta^T(X^T\Lambda^{-1}X/\sigma^2 + V_0^{-1})\beta\right)\right\} \end{aligned}$$

we recognize this as being proportional to a multivariate normal density, with

$$\beta|\{\lambda_i\}, \sigma^2, \mathcal{D} \sim N(\beta_1, V_1)$$
$$\beta_1 = V_1\left(X'\Lambda^{-1}y/\sigma^2 + V_0^{-1}\beta_0\right) \qquad V_1 = \left(X'\Lambda^{-1}X/\sigma^2 + V_0^{-1}\right)^{-1}$$

The σ^2 conditional distribution ($\sigma^2|\Lambda, \mathcal{D}, \beta$)

As in most normal sampling problems, the semiconjugate prior distribution for σ^2 is an inverse-gamma distribution. Letting $\gamma = 1/\sigma^2$ be the measurement precision, this implies that $\gamma \sim G\left(\frac{\nu_0}{2}, \frac{\nu_0\sigma_0^2}{2}\right)$ then

$$
\begin{aligned}
p(\gamma|\mathcal{D},\beta) &\propto p(\gamma)p(y|X,\beta,\gamma) \\
&\propto \left[\gamma^{\nu_0/2-1}\exp\left\{-\gamma\times\frac{\nu_0\sigma_0^2}{2}\right\}\right]\times\left[\gamma^{\frac{n}{2}}\exp\left\{-\frac{\gamma}{2}\sum_{i=1}^{n}\hat{\epsilon}_i^2\right\}\right] \\
&\propto \gamma^{(\nu_0+n)/2-1}\exp\left\{-\gamma\left[\nu_0\sigma_0^2+\sum_{i=1}^{n}\hat{\epsilon}_i^2\right]/2\right\}
\end{aligned}
$$

which we recognize as a gamma density, so that

$$
\sigma^2|\Lambda,\mathcal{D},\beta\sim IG\left(\frac{\nu_0+n}{2},\frac{\nu_0\sigma_0^2+\sum_{i=1}^{n}\hat{\epsilon}_i^2}{2}\right)
$$

Recall that $\sum_{i=1}^{n}\hat{\epsilon}_i^2=(y-X\beta)^T\Lambda^{-1}(y-x\beta)$

$$
\begin{aligned}
\sigma^2|\Lambda,\mathcal{D},\beta &\sim IG\left(\frac{\nu_0+n}{2},\frac{\nu_0\sigma_0^2+(y-X\beta)^T\Lambda^{-1}(y-x\beta)}{2}\right) \\
\sigma^2|\Lambda,\mathcal{D},\beta &\sim IG\left(\frac{\nu_1}{2},\frac{\nu_1\sigma_1^2}{2}\right) \\
\nu_1=\nu_0+n &\qquad \nu_1\sigma_1^2=\nu_0\sigma_0^2+(y-X\beta)^T\Lambda^{-1}(y-x\beta)
\end{aligned}
$$

The λ_i conditional distribution ($\lambda_i|\beta,\sigma^2,\mathcal{D}$)

Lastly we have that

$$
\begin{aligned}
p\left(\lambda_i|\beta,\sigma^2,y_i,x_i\right) &\propto p\left(y_i|\lambda_i,\beta,\sigma^2,x_i\right)p(\lambda_i) \\
p\left(\lambda_i|\beta,\sigma^2,y_i,x_i\right) &\propto \frac{1}{\sqrt{2\pi\sigma^2\lambda}}\exp\left\{-\frac{1}{2}\left(\left(y_i-x_i^T\beta\right)^2\sigma^{-2}\lambda_i^{-1}\right)\right\}\exp\{-0.5\lambda_i\} \\
p\left(\lambda_i|\beta,\sigma^2,y_i,x_i\right) &\propto \lambda^{-1/2}\exp\{-0.5\lambda_i\}\exp\left\{-0.5\left(\left(\frac{y_i-x_i'\beta}{\sigma}\right)^2\lambda_i^{-1}\right)\right\} \\
p\left(\lambda_i|\beta,\sigma^2,y_i,x_i\right) &\propto \lambda^{-1/2}\exp\left\{-0.5\left(\lambda_i+\left(\frac{y_i-x_i'\beta}{\sigma}\right)^2\lambda_i^{-1}\right)\right\}
\end{aligned}
$$

We claim that this distribution is of the generalized inverse Gaussian (GIG) form. Following Shuster (1968), Michael, et. al. (1976), and Carlin and Polson (1991), we outline a strategy for obtaining a draw from this GIG density.

We say that x follows an inverse Gaussian distribution ($x\sim invGauss(\psi,\mu)$) if

$$
p(x|\psi,\mu)\propto x^{-3/2}\exp\left(-\frac{\psi(x-\mu)^2}{2x\mu^2}\right),\quad x>0
$$

Now, let $z=x^{-1}$. It follows by a change of variables that

$$p(z|\psi,\mu) \propto z^{-2} z^{3/2} \exp\left(-\frac{\psi\left(z^{-1}-\mu\right)^2}{2z^{-1}\mu^2}\right)$$
$$\propto z^{-1/2} \exp\left(-\frac{\psi}{2}\left[z+\mu^{-2}z^{-1}\right]\right)$$

Then notice that the posterior conditional for λ_i, follows that the reciprocal of an $invGauss(1, |\sigma/(y_i - x_i\beta)|)$. **This means that a draw of λ_i can be done by inverting a draw from the inverse Gaussian distribution.** Then, the only step is to draw from the inverse Gaussian distribution.

Shuster (1968) notes that if x has the inverse Gaussian density, then $\psi(x-\mu)^2/x\mu^2 \sim \chi^2(1)$, a chi-square distribution with one degree of freedom. Let $\nu_2 = \psi(x-\mu)^2/x\mu^2$, them the roots of nu_2, denoted here as x_1 and x_2 are obtained as

$$x_1 = \mu + \frac{\mu^2\nu_2}{2\psi} - \frac{\mu}{2\psi}\sqrt{4\mu\psi\nu_2 + \mu^2\nu_2^2}$$
$$x_2 = \mu^2/x_1$$

Michael et al. (1976) use this idea to show that one can obtain a draw from the inverse Gaussian (ψ,μ) density by first drawing $\nu_2 \sim \chi^2(1)$, calculating the roots x_1 and x_2 from the preceding equations, and then setting x equal to x_1 with probability $\mu/(\mu+x_1)$ and equal to x2 with probability $x_1/(\mu+x_1)$.

PART C)

We now simulate $n = 200$ observations from the above linear regression with double exponential errors model, where $\beta = (0,1,2,3)'$, $\sigma^2 = 1$ and $x_{ij} \sim N(0,1)$. Afther the simulation we implement the above MCMC scheme and produce posterior summaries of the main parameters. We also try to answer if the simple MH algorithm to sample λ_i be reasonable for your simulated data or the full-fledge Gibbs sampler performs better.

We first load libraries and clear any variables to start with a clear enviroment.

```
# Libraries
library(statmod) # Used for Inverse Gausian
library(nimble)  # Used for double exponential
library(pracma)  # Used for Time Evaluation

# Clear Vars
rm(list = ls())
```

For the Full metropolis hasting we will need to determine the proposal functions and likelihood functions.

```
func_q = function(lambda_i, y_i, x_i, beta, sigma)
{
  ret = -0.5*log(lambda_i) -0.5*(lambda_i+( (y_i - x_i %*% beta)/sigma  )^2 * 1/lambda_i)
  return(ret)
}

func_F = function(lambda_i, y_i, x_i, beta, sig2)
{
  erro = y_i - x_i%*%beta
  ret = dnorm(erro, mean = 0, sd = (sig2*lambda_i)^0.5, log = TRUE) - lambda_i/2
  return(ret)
}
```

We now start with the initial setup. We fix a seed por replication porposes, and set the initial values of $X = \{x_{i1}, x_{i2}, x_{i3}, x_{i4}\}_{i=1}^{n}$

```
# Replication Seed
set.seed(12345)

# Sample size
n = 200

# regressors
nregress = 4
beta_true = as.matrix(0:(nregress-1), nregress,1)

# Regressors Matrix
X = matrix(rnorm(nregress*n,0,1), n, nregress)

# Declare of sigma
sigma_true = 1
```

We then we proced seting up the simulation. We use a draw form the double exponential for the erros, however a draw for lambda and then the construction of the errors could also be used.

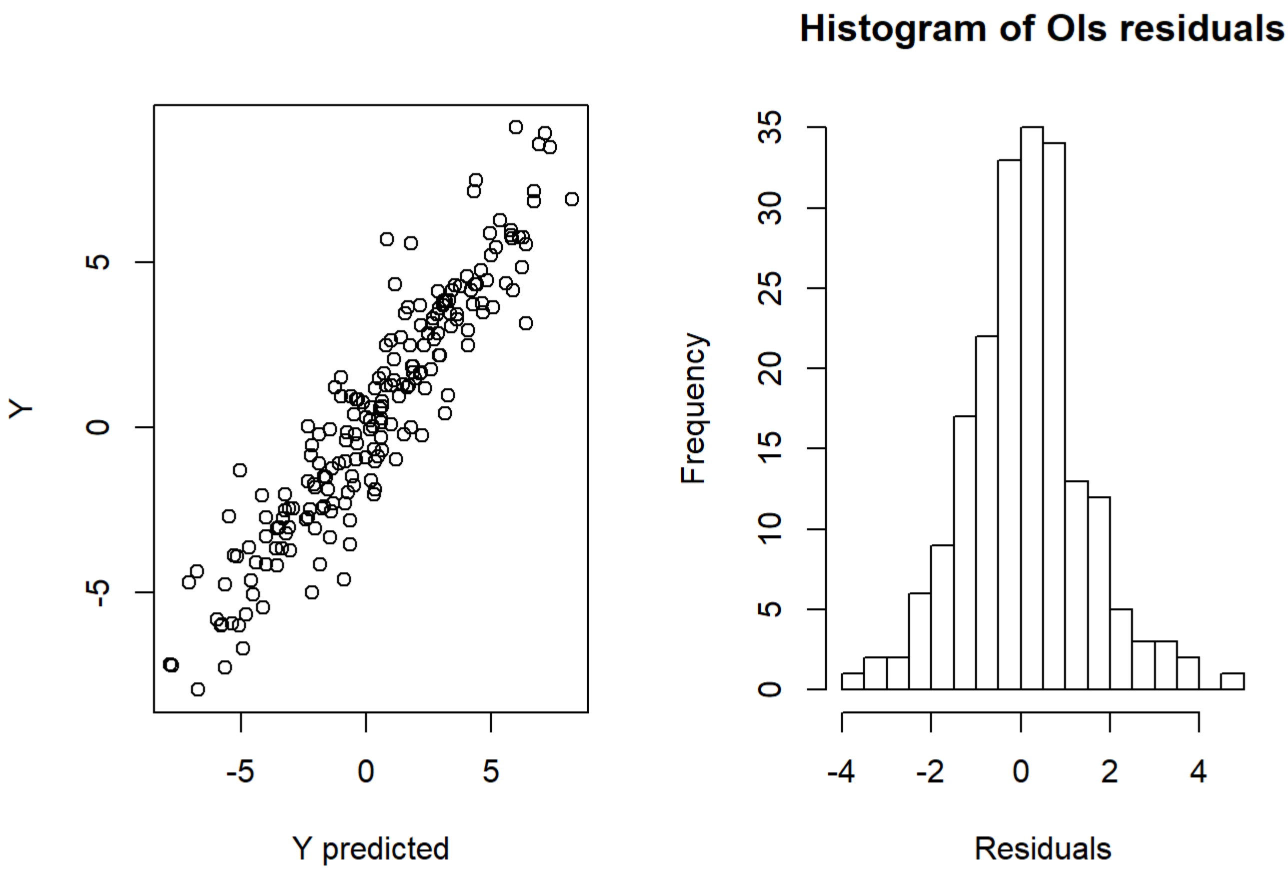
```
# We opt to use the Double exponential extrection
USE_DOUBLE_EXP = TRUE

# Simulacao
if(USE_DOUBLE_EXP){
  error_true = nimble::rdexp(n, location = 0, scale = sigma_true)
} else
{
  lambda_true = rexp(n,1/2)
  error_true  = rnorm(n, 0, sqrt(sigma_true*lambda_true))
}

# Simulation
Y <- X%*%beta_true + error_true
```

Ouw first step is to look at the classical OLS estimator.

```
# Ols model
ols = lm(Y ~ X -1);
sigma2.ols = summary(ols)$sigma^2
beta.ols = matrix(ols$coefficients, nregress, 1)
summary(ols)
```

```
##
## Call:
## lm(formula = Y ~ X - 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6968 -0.7314  0.1168  0.7823  4.7151
##
## Coefficients:
##    Estimate Std. Error t value Pr(>|t|)
## X1 -0.03397    0.08843  -0.384    0.701
## X2  0.86260    0.09895   8.718  1.2e-15 ***
## X3  1.96746    0.09744  20.191  < 2e-16 ***
## X4  2.93288    0.09408  31.175  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.338 on 196 degrees of freedom
## Multiple R-squared:  0.8689, Adjusted R-squared:  0.8663
## F-statistic: 324.9 on 4 and 196 DF,  p-value: < 2.2e-16
```

```
# histogram
par(mfrow=c(1,2))
plot(X%*%beta_true, Y, xlab = "Y predicted", ylab = "Y")
hist(ols$residuals, breaks=15, main="Histogram of Ols residuals", xlab = "Residuals", ylab = "Fr
equency")
```



Now we make the initial setup of the MCMC. We set a *Burn up* sample of size 10^4, and a final sample of size 10^3. We also store the draws in a table.

```
# MCMC set-up
M0    = 1000  # Final
M     = 10000 # Burn up
niter = M0+M


# TABLE DRAWS
ncol.draws = 1 + nregress
draws.mc = matrix(0, nrow = niter, ncol = ncol.draws)
colnames(draws.mc) = c("sigma", paste("beta", 1:nregress))

draws.gibbs = matrix(0, nrow = niter, ncol = ncol.draws)
colnames(draws.gibbs) = c("sigma", paste("beta", 1:nregress))
```

For the Priors we will set the following. we will assume that $\beta_0 = 0$, this impplies that we do not expect Y to be correlated with X, however we are insecure about this fact and assume a standard deviation of 5 for each beta ($V_0 = diag(25)_{n=4}$). For σ^2 we will set the priors $\sigma_0 = 1$ and $\nu_0 = 2.5$. We also set the initial values for our interations.

```
# priors of beta
beta_0 = matrix(0, nregress, 1)
V_0 = diag(25, nregress)

# priors of sigma
sigma2_0 = 1
nu_0 = 2.5

# initial Values
sigma2 = sigma2.ols
beta = beta.ols
lambda = rep(1,n)
```

Then we proceed with the MCMC, one could make the draw from the inverse Gaussian using the *statmod* r package or the procedure done by michael et. al.

```
# We opt to use Michael method
USE_MICHAEL_METHOD = TRUE

# table to store execution time
dt_time = data.frame(Method = c("MH", "GIBBS"), Time = NA, ess=NA, ess_ps = NA)
for (k in 1:2) {
  if(k==1)
  { MH = TRUE }
  else
  { MH=FALSE }

  pracma::tic()
  for (i in 1:(niter)){

    # Inicialize Lambda^{-1}  matrix
    Lambda_1 = solve(diag(lambda))

    # full conditional of sigma2
    d0=(nu_0 * sigma2_0)/2
    par1 = (nu_0 + n)/2
    par2 = d0 + ( t(Y-X%*%beta) %*% Lambda_1 %*% (Y-X%*%beta) )/2

    # Conditional distribution of sigma
    sig2 = 1/rgamma(1, par1, par2)

    # full conditional of beta
    XtX = t(X) %*% Lambda_1 %*% X
    XtY = t(X) %*% Lambda_1 %*% Y

    V_1   = solve(XtX/sig2 + solve(V_0))
    beta_1 = V_1 %*% (XtY/sig2 + solve(V_0) %*% beta_0)
    beta =  beta_1 + t(chol(V_1)) %*% rnorm(nregress)

    # Foolowing Koop. (Bayesian Econometrics Methods) pag 260

    # Draw of lambda
    for (j in 1:n){
      y_j = Y[j, 1]
      x_j = X[j, ]

      # draw de nu_0
      nu_michael_0 = rchisq(1,1)

      # mu for row j
      mu_j = abs(sqrt(sig2)/(y_j - x_j %*% beta))

      if(USE_MICHAEL_METHOD){
        #  x1 e x2
        x_1 = mu_j + (mu_j^2 * nu_michael_0)/2 - mu_j/2 * (4 * mu_j * nu_michael_0 + mu_j^2 * nu
_michael_0^2)^0.5
        x_2 = mu_j^2 / x_1

        # decide between x_1 and x_2
```

```
        p.treshold = mu_j/(mu_j + x_1)
        if (runif(1) < p.treshold)
        {
          x_star = x_1
        }
        else
        {
          x_star = x_2
        }

        # invert x_star
        lambda_j = 1/x_star
      }
      else
      {
        lambda_j = statmod::rinvgauss(1, mu_j, shape = 1)
      }


      # now the Metropolis-Hasting
      if ((lambda_j >0) & (MH)){

        deno = func_F(lambda_j, y_j, x_j, beta, sig2) + func_q(lambda[j], y_j, x_j, beta, sqrt(s
ig2))
        nume = func_F(lambda[j], y_j, x_j, beta, sig2) + func_q(lambda_j, y_j, x_j, beta, sqrt(s
ig2))

        log.rho = min(0, nume-deno)
        if (log(runif(1)) < log.rho){
          lambda[j] = lambda_j
        }
      }
    }

    # storing draws
    if(MH)
    {draws.mc[i,] = c(sig2, beta)}
    else
    {draws.gibbs[i,] = c(sig2, beta)}

  }

  # Determine Execution Time
  dt_time[k, "Time"] = pracma::toc()

  # Determine ESS
  if(MH)
  {
    dt_time[k, "ess"] = round(M/(1+2*sum(acf(draws.mc[,"sigma"],lag.max=1000,plot=FALSE)$acf[2:1
001])))
  }
  else
  {
    dt_time[k, "ess"] = round(M/(1+2*sum(acf(draws.gibbs[,"sigma"],lag.max=1000,plot=FALSE)$acf[
```

```
2:1001])))
  }

}
```

```
## elapsed time is 78.940000 seconds
## elapsed time is 44.750000 seconds
```

```
# Determine the ess per second
dt_time$ess_ps = dt_time$ess / dt_time$Time
```

We them print the distributions

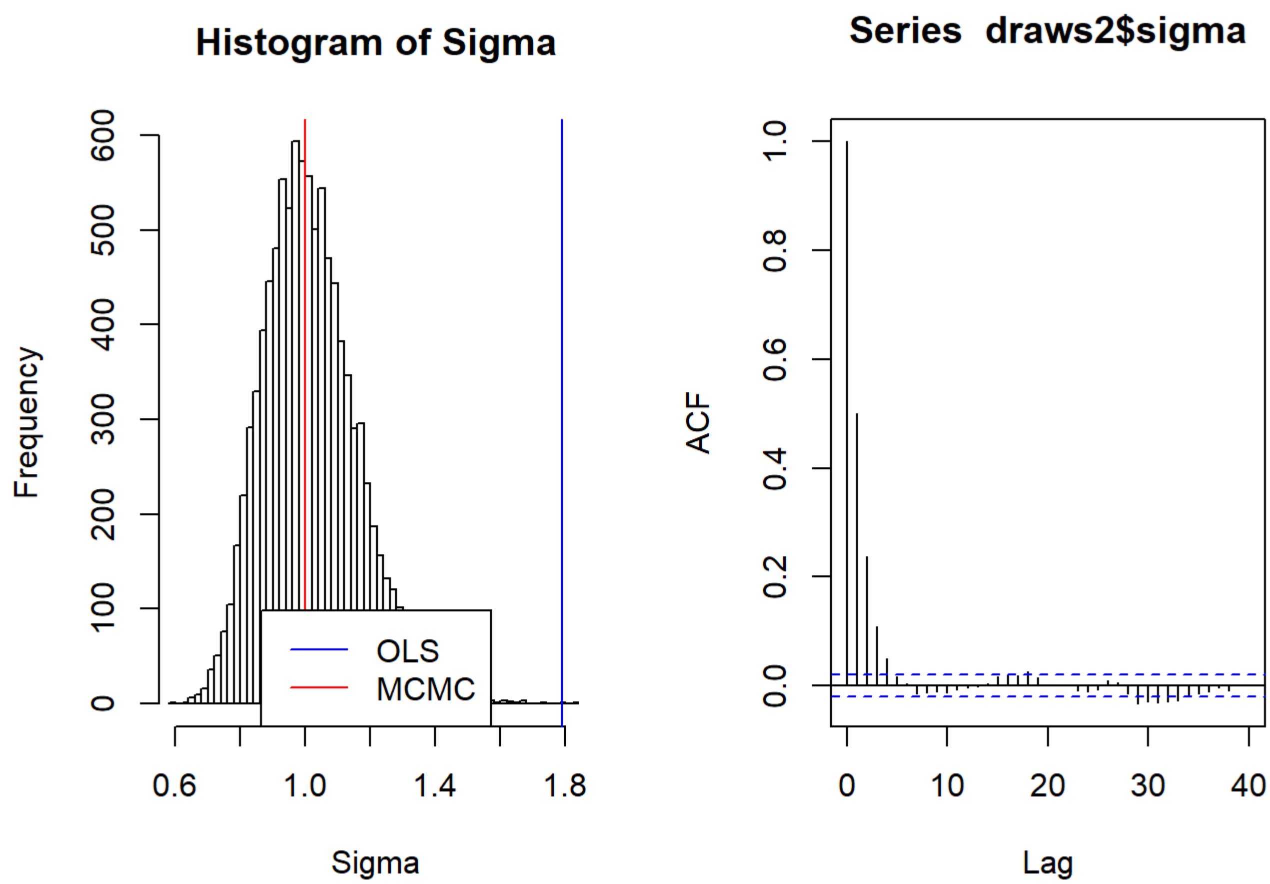
```
draws2 = data.frame(draws.mc[(M0+1):niter,])
colnames(draws2)
```

```
## [1] "sigma"  "beta.1" "beta.2" "beta.3" "beta.4"
```

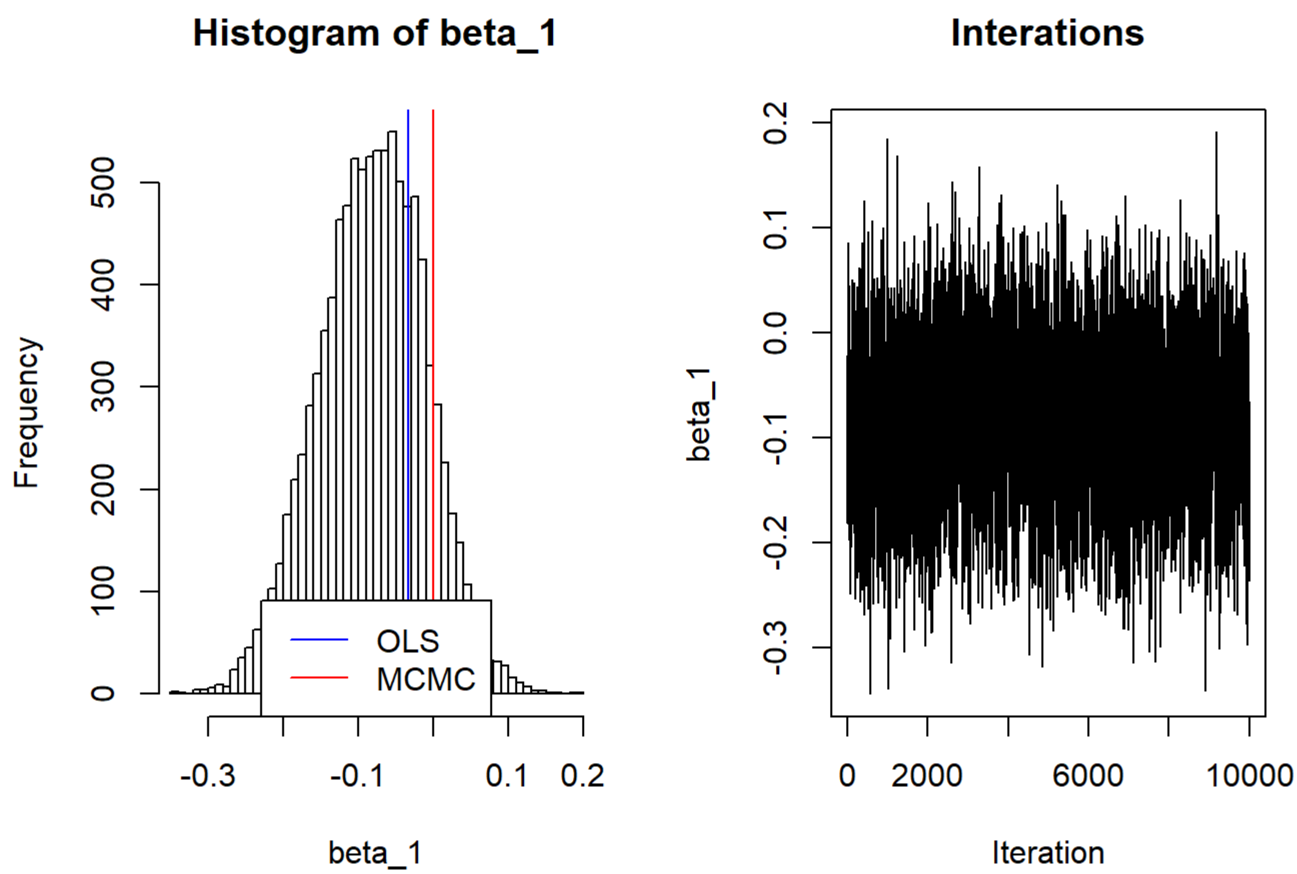
```
summary(draws2$sigma)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5986  0.9145  1.0045  1.0172  1.1057  1.8359
```

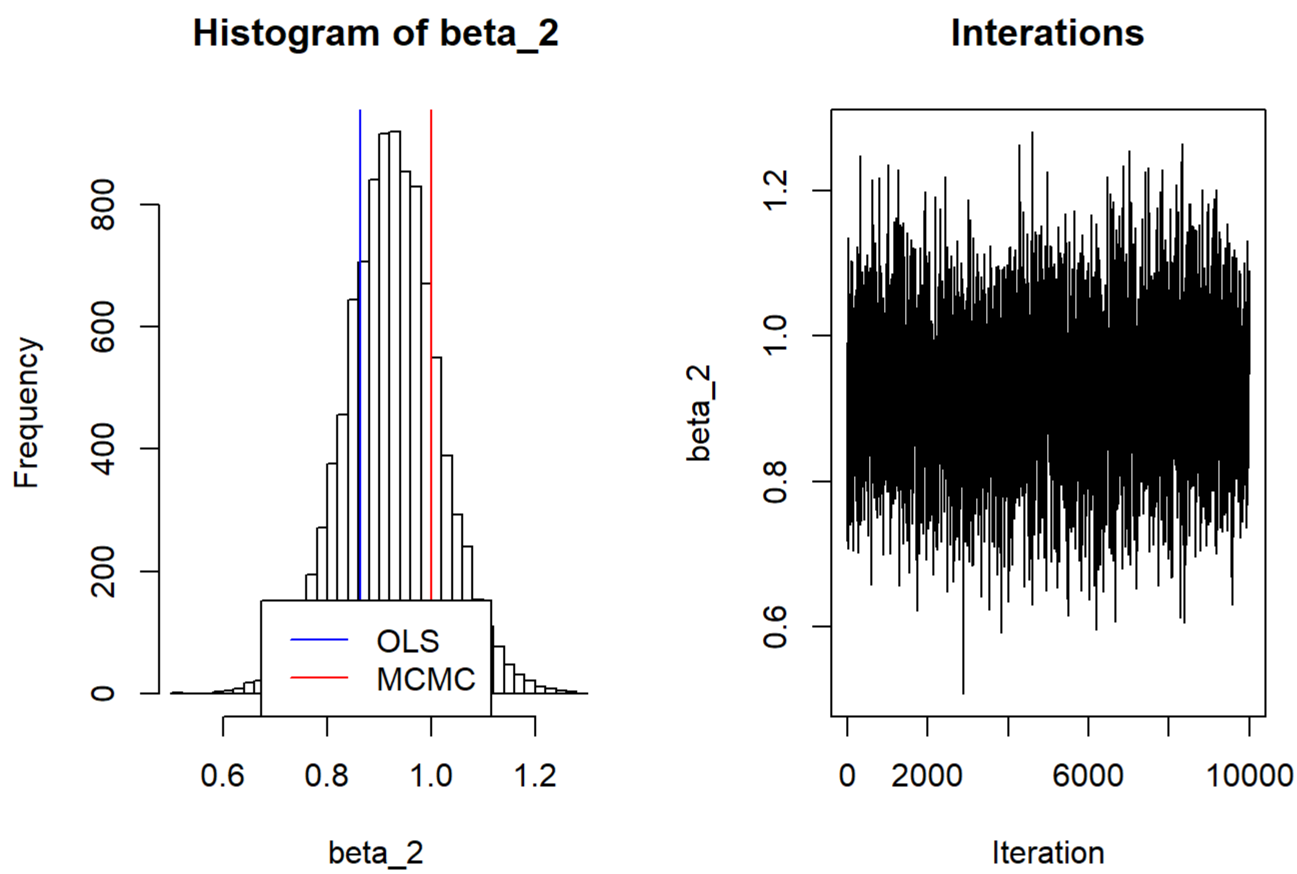
```
par(mfrow=c(1,2))
hist(draws2$sigma, breaks = 50, main="Histogram of Sigma", xlab = "Sigma", ylab = "Frequency", x
lim = range(sigma_true, sigma2.ols, draws2$sigma))
abline(v=sigma_true, col="red")
abline(v=sigma2.ols, col="blue")
legend("bottom", c("OLS", "MCMC"), col = c("blue", "red"), lty=1)
acf(draws2$sigma)
```



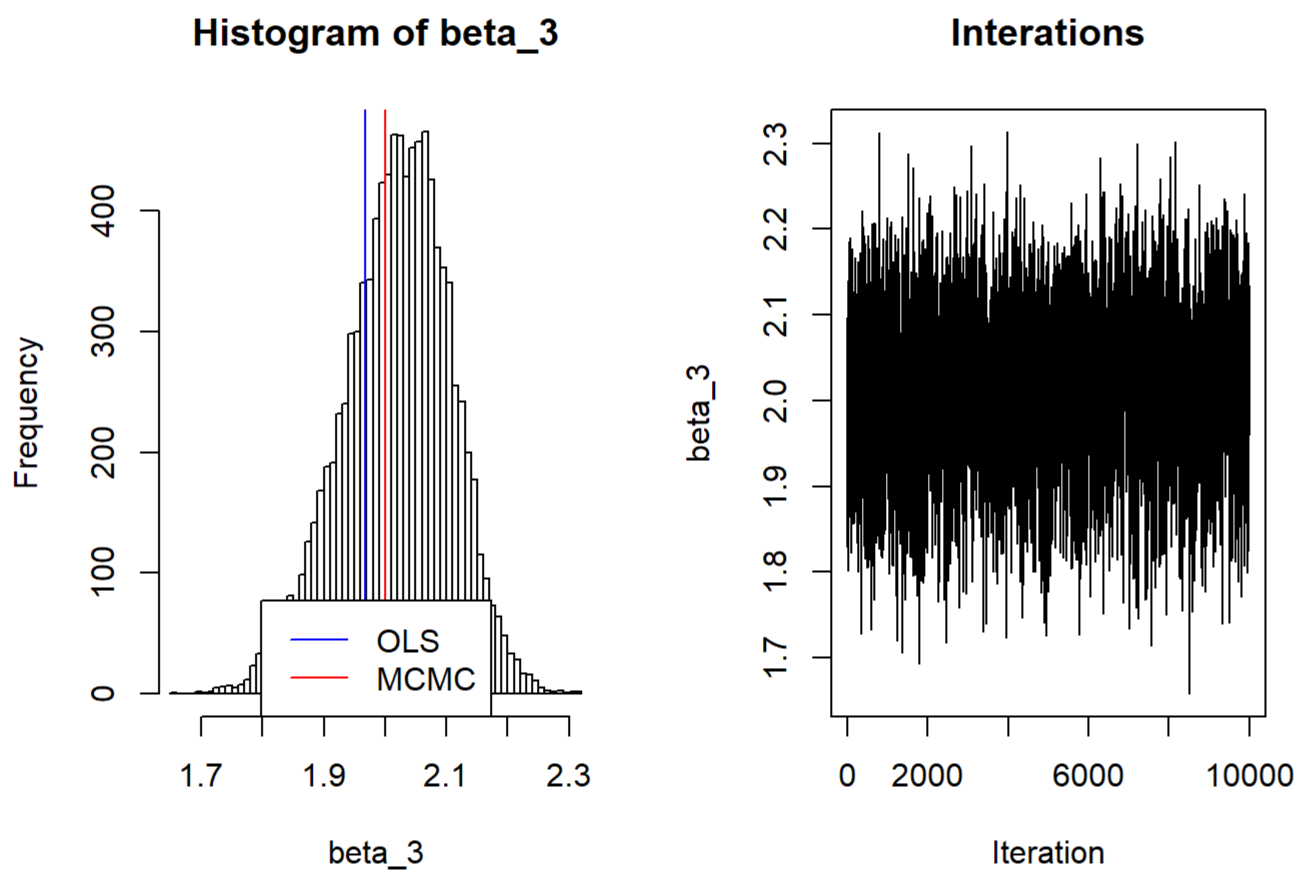

```
par(mfrow=c(1,2))
hist(draws2$beta.1, breaks = 50, main="Histogram of beta_1", xlab = "beta_1", ylab = "Frequency"
, xlim = range(beta_true[1], beta.ols[1], draws2$beta.1))
abline(v = beta_true[1], col="red")
abline(v = beta.ols[1], col="blue")
legend("bottom", c("OLS", "MCMC"), col = c("blue", "red"), lty=1)
plot(draws2$beta.1, xlab="Iteration", ylab="beta_1",main="Interations", type = "l")
```



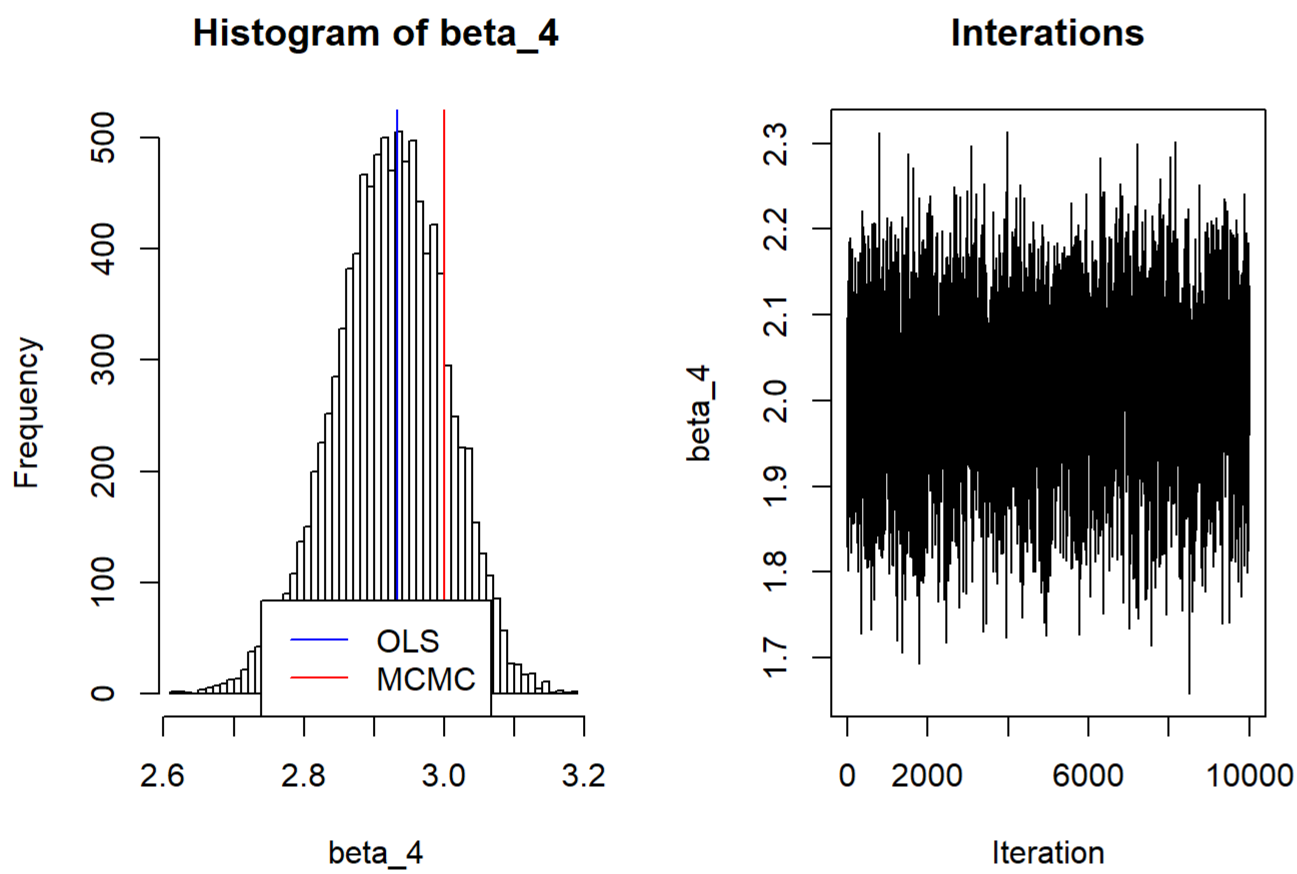

```
par(mfrow=c(1,2))
hist(draws2$beta.2, breaks = 50, main="Histogram of beta_2", xlab = "beta_2", ylab = "Frequency"
, xlim = range(beta_true[2], beta.ols[2], draws2$beta.2))
abline(v = beta_true[2], col="red")
abline(v = ols$coefficients[2], col="blue")
legend("bottom", c("OLS", "MCMC"), col = c("blue", "red"), lty=1)
plot(draws2$beta.2, xlab="Iteration", ylab="beta_2",main="Interations", type = "l")
```




```
par(mfrow=c(1,2))
hist(draws2$beta.3, breaks = 50, main="Histogram of beta_3", xlab = "beta_3", ylab = "Frequency"
, xlim = range(beta_true[3], beta.ols[3], draws2$beta.3))
abline(v = beta_true[3], col="red")
abline(v = ols$coefficients[3], col="blue")
legend("bottom", c("OLS", "MCMC"), col = c("blue", "red"), lty=1)
plot(draws2$beta.3, xlab="Iteration", ylab="beta_3",main="Interations", type = "l")
```




```
par(mfrow=c(1,2))
hist(draws2$beta.4, breaks = 50, main="Histogram of beta_4", xlab = "beta_4", ylab = "Frequency"
, xlim = range(beta_true[4], beta.ols[4], draws2$beta.4))
abline(v = beta_true[4], col="red")
abline(v = ols$coefficients[4], col="blue")
legend("bottom", c("OLS", "MCMC"), col = c("blue", "red"), lty=1)
plot(draws2$beta.3, xlab="Iteration", ylab="beta_4",main="Interations", type = "l")
```



Lastily we evaluate if MH perform better. We present the following table.

```
dt_time
```

```
##   Method  Time   ess   ess_ps
## 1     MH 78.94 46475 588.7383
## 2  GIBBS 44.75 37775 844.1341
```