Take Home

Bruno do Prado Costa Levy

February, 14th, 2020

Part A

$$y_i = x_i'\beta + \epsilon_i, \quad \epsilon_i|\lambda_i, \sigma^2 \sim N\left(0, \lambda_i\sigma^2\right)$$

where

$$\lambda_1, \ldots, \lambda_n \sim Exp(0.5)$$

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

Define $\psi_i = \lambda_i^{1/2}$, then:

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

Applying Hint 1 (from Andrews and Mallows (1974)):

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

Part B

One way to solve (simulate) from the posterior is to find the full conditionals for β, σ^2 and λ_i $(i = 1, ..., n)$. In order to sample from them, the procedure is similar to what we have made in class. In the case of the scale-mixing variables λ s, we can use a Metropolis Hastings step within the Gibbs sampler to draw them from the full conditional or sample directly from a Generalized Inverse Gaussian in a full-fledge Gibbs Sampler. The basic idea of the algorithms is to cycle across the condiditionals such that the distributions will converge to the joint distribution. We initialize the process with initial values for the hyperparameters of β, σ and λ.

$$p\left(\beta, \lambda_1, ..., \lambda_n, \sigma^2|y\right) \propto \left[\prod_{i=1}^n \phi\left(y_i; x_i\beta, \lambda_i\sigma^2\right) p\left(\lambda_i\right)\right] p(\beta) p\left(\sigma^2\right)$$

$$\propto det(\sigma_0^{-1} I)\Lambda^{-1/2} \exp\{-1/2\sigma_0^2[(y - X\beta)'\Lambda^{-1}(y - X\beta)]\} \times \prod_{i=1}^n 2\exp\{-\lambda_i/2\} \times \exp\{-1/2[\beta' V_0^{-1}\beta - 2\beta' V_0^{-1}\beta_0]\} \times$$

$$\times \sigma_0^{-\nu_0/2+1} \exp\{-\nu_0\sigma_0^2/2\sigma_0^2\}$$

Full Conditional for β:

$$(\beta|\{\lambda_i\},\sigma^2,y) \propto \exp\{\frac{-1}{2\sigma^2}[\beta' X'\Lambda^{-1}X\beta - 2\beta' X'\Lambda^{-1}y] - 1/2[\beta' V_0^{-1}\beta - 2\beta' V_0^{-1}\beta_0]\}$$

$$\propto \exp\{\beta'(\frac{X'\Lambda^{-1}y}{\sigma^2} + V_0^{-1}\beta_0) - 1/2[\beta'(\frac{X'\Lambda^{-1}X}{\sigma^2} + V_0^{-1})\beta]\}$$

$$\propto \exp\{-1/2(\frac{X'\Lambda^{-1}X}{\sigma^2} + V_0^{-1})[\beta'\beta - 2(\frac{X'\Lambda^{-1}X}{\sigma^2} + V_0^{-1})^{-1}(\frac{X'\Lambda^{-1}y}{\sigma^2} + V_0^{-1}\beta_0)\}$$

$$p(\beta|\{\lambda_i\},\sigma^2,y) \sim N([\frac{X'\Lambda^{-1}X}{\sigma^2} + V_0^{-1})^{-1}(\frac{X'\Lambda^{-1}y}{\sigma^2} + V_0^{-1}\beta_0)], \quad (\frac{X'\Lambda^{-1}X}{\sigma^2} + V_0^{-1})^{-1})$$

Full Condidional for σ:

$$p(\sigma^2|\{\lambda_i\},\beta_i,y) \propto \det(\sigma_0^{-1}I)\exp\{-1/2\sigma_0^2[(y-X\beta)'\Lambda^{-1}(y-X\beta)]\} \times (\sigma_0^{-\nu/2+1})\exp\{-\nu_0\sigma_0^2/2\sigma_0^2\}$$

$$\propto (\sigma_0^2)^{-(\nu_0/2+n/2)+1}\exp\{-1/2\sigma_0^2[\nu_0\sigma_0^2 + (y-X\beta)'\Lambda^{-1}(y-X\beta)]\}$$

$$p(\sigma^2|\{\lambda_i\},\beta_i,y) \sim IG(\nu_1/2, \sigma_1^2\nu_1/2)$$

where

$$\nu_1 = \nu_0 + n$$

and

$$\nu_1\sigma_1 = \nu_0\sigma_0 + (y-X\beta)'\Lambda^{-1}(y-X\beta)$$

Full Conditional for λ_i:

$$p(\lambda_i|\beta,\sigma,\lambda_{-i},y) \propto 2\exp\{-\lambda_i/2\}\lambda_i^{-1/2}\exp\{\frac{-(y_i - x_i\beta)^2/2\sigma^2}{\lambda_i}\}$$

$$\propto \lambda_i^{-1/2}\exp\{-1/2[\lambda_i + (\frac{y_i - x_i\beta}{\sigma})\lambda_i^{-1}]\}$$

PSEUDO - CODE

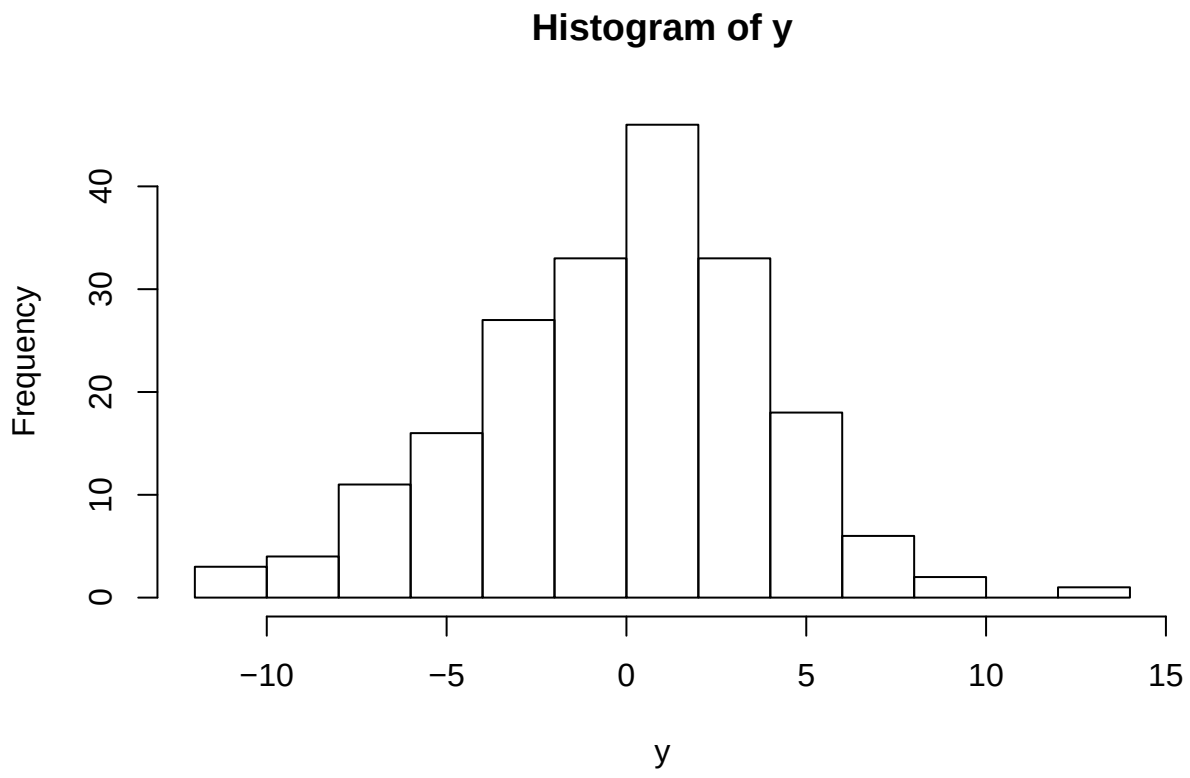
Given a current state of the parameters $\theta^{(s)} = \{\beta^{(s)}, \sigma^{2(s)}, \lambda^{2(s)}\}$ we generate a new state as follows:

1. Sample $\beta^{(s+1)} \sim p(\beta|\sigma^{2(s)}, \{\lambda^{(s)}\}, D_n)$
2. Sample $\sigma^{2(s+1)} \sim p(\sigma^2|\beta^{(s+1)}, \{\lambda^{(s)}\}, D_n)$
3. Sample $\lambda^{(s+1)} \sim p(\lambda|\sigma^{2(s+1)}, \beta^{(s+1)}, D_n)$

PART C

Simulating the data

```
n      = 200
beta.t = c(0,1,2,3)
sig.t = 1

# Sampling Lambda from an exponential and plugging on the variance of the Normal

set.seed(1234)
X = cbind(1,rnorm(n),rnorm(n),rnorm(n))
y = rep(0,n)
for (i in 1:n){
  lambda = rexp(1,0.5)
  y[i] = beta.t%*%X[i,] + rnorm(1,0,sig.t*sqrt(lambda))
}

hist(y)
```

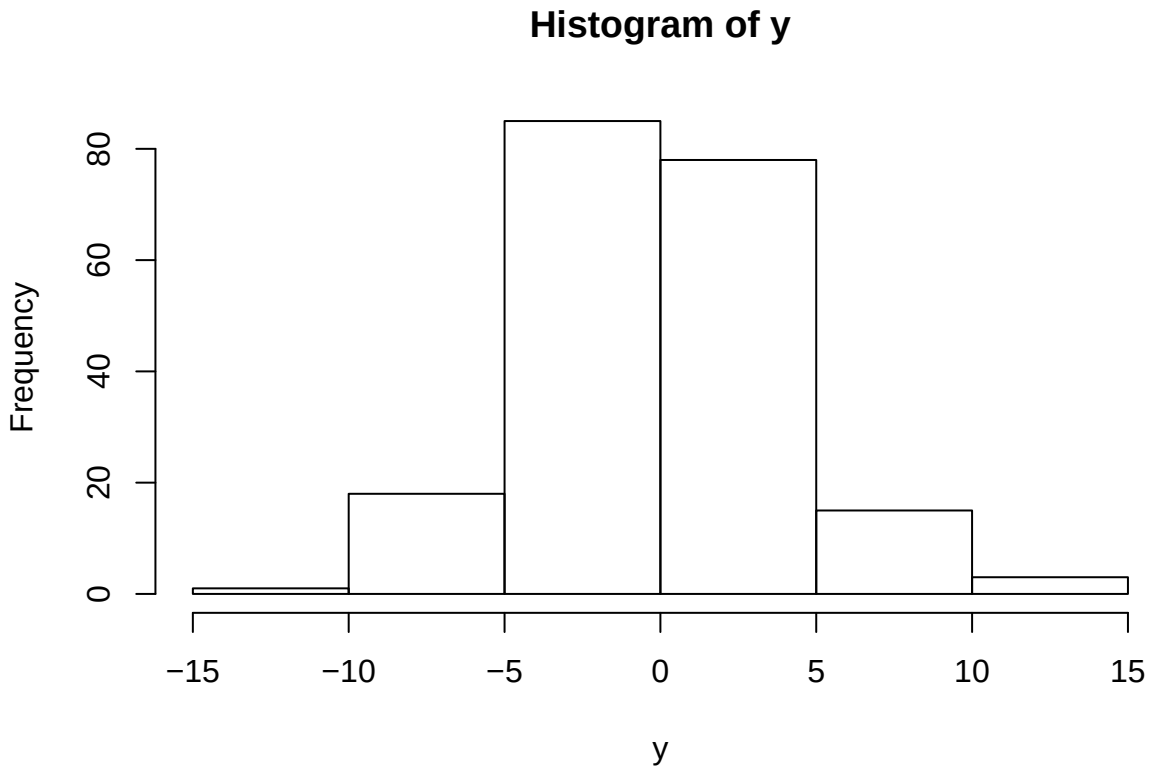



```
# Other way: sampling directly from a double exponential

library(smoothmest)
```

```
## Warning: package 'smoothmest' was built under R version 3.5.3
```

```
## Loading required package: MASS
```

```
y = rep(0,n)
for (i in 1:n){
  y[i] = beta.t%*%X[i,] + rdoublex(1,mu=0,lambda=1)
}


hist(y)
```



Full-fledge Gibss Sampler

```
# Generator for the Generalized Inverse Gaussian (GIG) distribution
library(GIGrvg)
```

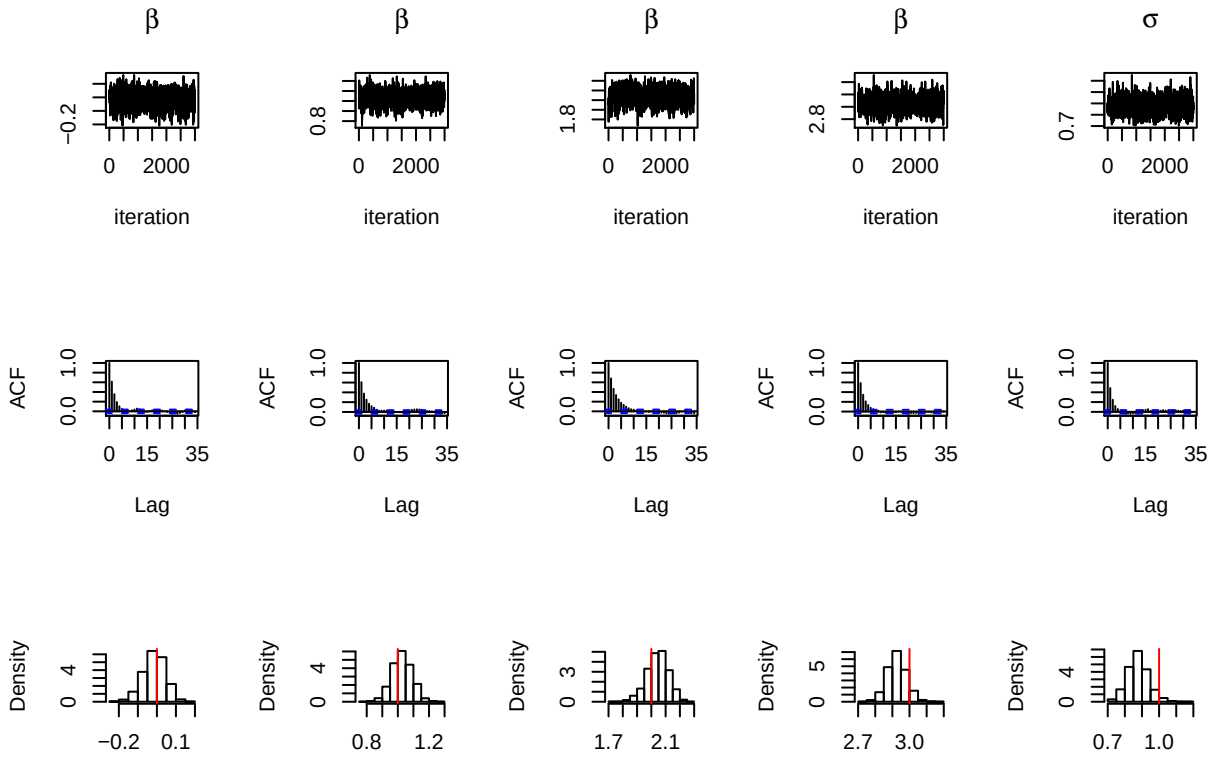
```
## Warning: package 'GIGrvg' was built under R version 3.5.2
```

```
# Prior hyperparameters
iV0 = diag(1/3,4)
b0 = betas = c(0,0,0,0)
lambda = rep(1,n)
nu0 = 5
sig0 =1
```

```r
# MCMC set-up
sigma2 = 1
M0     = 2000
M      = 3000
niter = M0+M
draws1 = matrix(0,niter,5)

# MCMC algorithm
gibbs.time = system.time({
for (iter in 1:niter){

  # Sampling the lambda's
    # Individually

   lambda = rep(0,n)
   for (i in 1:n){
     chi = (y[i]-X[i,]%*%betas)^2/sigma2

     lambda[i] =rgig(n=1, lambda = 0.5, chi = chi, psi= 1)
   }

    # Jointly

  #chi = t((y-X%*%betas))%*%(y-X%*%betas)/sigma2
  #lambda = rgig(n=n, lambda = 0.5, chi = chi, psi= 1)


  ilambda = solve(diag(lambda))


  # Sampling betas

  V1 = solve(t(X)%*%ilambda%*%X/sigma2+iV0)
  b1 = V1%*%t(X)%*%ilambda%*%y/sigma2
  betas = b1 + t(chol(V1))%*%rnorm(4)

  # Sampling sigma2
  nu1 = nu0 + n
  s1 = nu0*sig0 + t((y-X%*%betas))%*%ilambda%*%(y-X%*%betas)
  sigma2 = 1/rgamma(1,nu1/2,s1/2)

  # Storing the results
  draws1[iter,] = c(betas,sqrt(sigma2))
}
draws1 = draws1[(M0+1):niter,]
})

gibbs.time = round(as.numeric(gibbs.time[3]),2)
```

Posterior summaries

```
par(mfrow=c(3,5))
ts.plot(draws1[,1],xlab="iteration",ylab="",main=expression(beta))
ts.plot(draws1[,2],xlab="iteration",ylab="",main=expression(beta))
ts.plot(draws1[,3],xlab="iteration",ylab="",main=expression(beta))
ts.plot(draws1[,4],xlab="iteration",ylab="",main=expression(beta))
ts.plot(draws1[,5],xlab="iteration",ylab="",main=expression(sigma))

acf(draws1[,1],main="")
acf(draws1[,2],main="")
acf(draws1[,3],main="")
acf(draws1[,4],main="")
acf(draws1[,5],main="")

hist(draws1[,1],prob=TRUE,main="",xlab="")
abline(v=beta.t[1],col=2)
hist(draws1[,2],prob=TRUE,main="",xlab="")
abline(v=beta.t[2],col=2)
hist(draws1[,3],prob=TRUE,main="",xlab="")
abline(v=beta.t[3],col=2)
hist(draws1[,4],prob=TRUE,main="",xlab="")
abline(v=beta.t[4],col=2)
hist(draws1[,5],prob=TRUE,main="",xlab="")
abline(v=sig.t,col=2)
```



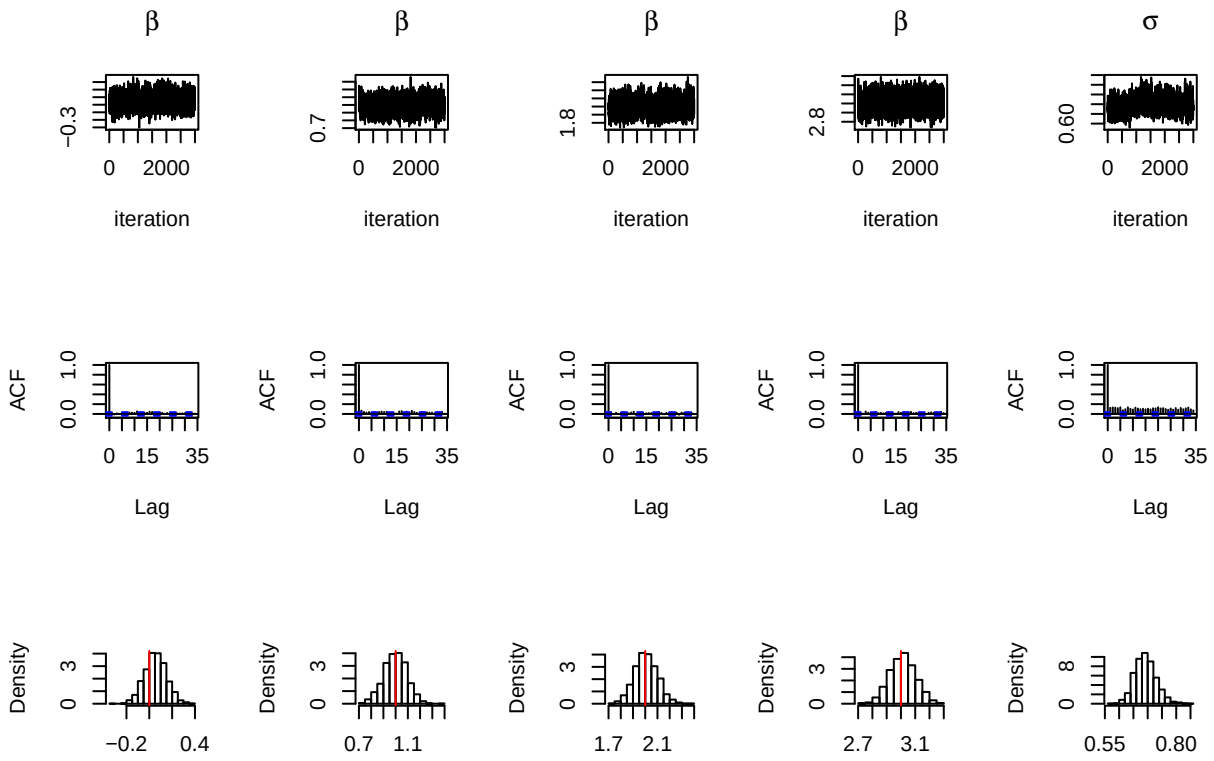
METROPOLIS HASTINGS

```
# Full Conditional for lambda

post_lambda = function(lambda, betas,sigma){
  -0.5*log(lambda) -0.5*(lambda + ( ((y-X%*%betas)/sigma)^2)/lambda )
}


lambda = rep(1,n)
sd.lambda = 1.5


# MCMC algorithm
iV0 = diag(1/3,4)
b0 = betas = c(0,0,0,0)
lambda = rep(1,n)
nu0 = 5
sig0 =1
sd.lambda = 1


MH.time = system.time({
draws2 = matrix(0,niter,5)
for (iter in 1:niter){

  ilambda = solve(diag(lambda))

  # Sampling betas

  V1 = solve(t(X)%*%ilambda%*%X/sigma2+iV0)
  b1 = V1%*%t(X)%*%ilambda%*%y/sigma2
  betas = b1 + t(chol(V1))%*%rnorm(4)

  # Sampling sigma2
  nu1 = nu0 + n
  s1 = nu0*sig0 + t((y-X%*%betas))%*%ilambda%*%(y-X%*%betas)
  sigma2 = 1/rgamma(1,nu1/2,s1/2)


  ## Metropolis within Gibbs

  # Full conditional of lambda
  for (j in 1:n){
    lambda1 = rlnorm(1,lambda[j],sd.lambda)
    if (lambda1>0){
      nume = post_lambda(lambda1,betas,sqrt(sigma2))-dlnorm(lambda1,lambda[j],sd.lambda,log=TRUE)
      deno = post_lambda(lambda[j],betas,sqrt(sigma2))-dlnorm(lambda[j],lambda1,sd.lambda,log=TRUE)
      log.alpha = min(0,nume-deno)

      if (log(runif(1))<log.alpha){
        lambda[j] = lambda1
      }
```

```
    }
  }

  # Storing the results
  draws2[iter,] = c(betas,sqrt(sigma2))
}
draws2 = draws2[(M0+1):niter,]
})

MH.time = round(as.numeric(MH.time[3]),2)
```

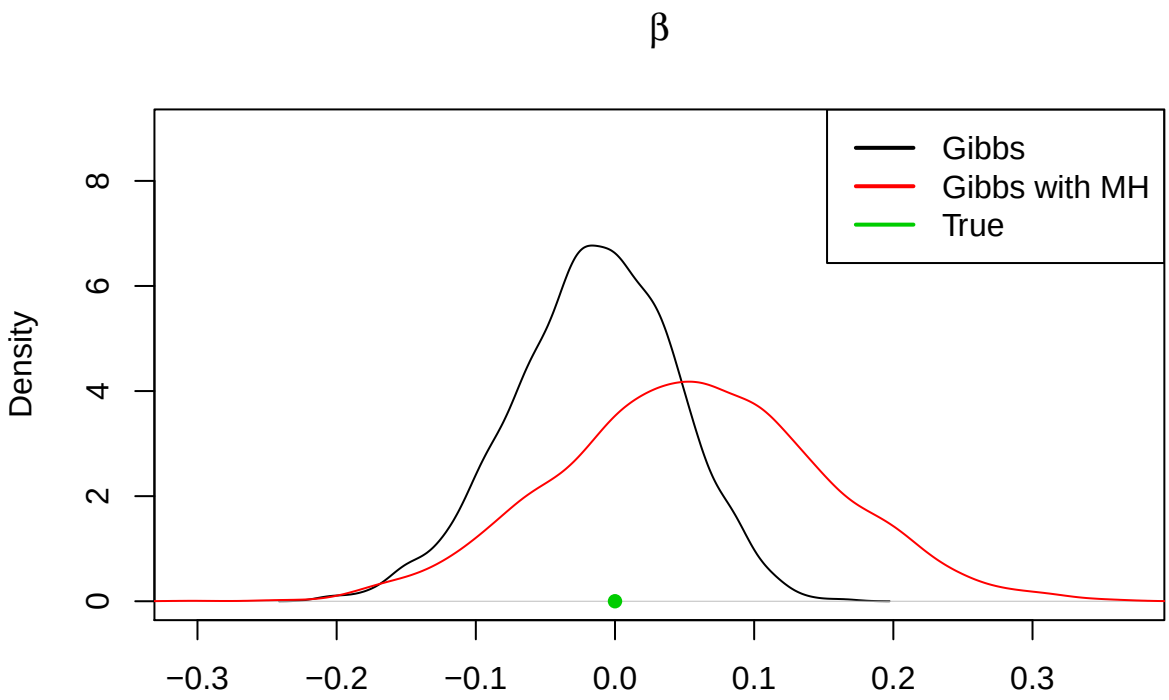
Posterior summaries

```
par(mfrow=c(3,5))
ts.plot(draws2[,1],xlab="iteration",ylab="",main=expression(beta))
ts.plot(draws2[,2],xlab="iteration",ylab="",main=expression(beta))
ts.plot(draws2[,3],xlab="iteration",ylab="",main=expression(beta))
ts.plot(draws2[,4],xlab="iteration",ylab="",main=expression(beta))
ts.plot(draws2[,5],xlab="iteration",ylab="",main=expression(sigma))

acf(draws2[,1],main="")
acf(draws2[,2],main="")
acf(draws2[,3],main="")
acf(draws2[,4],main="")
acf(draws2[,5],main="")

hist(draws2[,1],prob=TRUE,main="",xlab="")
abline(v=beta.t[1],col=2)
hist(draws2[,2],prob=TRUE,main="",xlab="")
abline(v=beta.t[2],col=2)
hist(draws2[,3],prob=TRUE,main="",xlab="")
abline(v=beta.t[3],col=2)
hist(draws2[,4],prob=TRUE,main="",xlab="")
abline(v=beta.t[4],col=2)
hist(draws2[,5],prob=TRUE,main="",xlab="")
abline(v=sig.t,col=2)
```

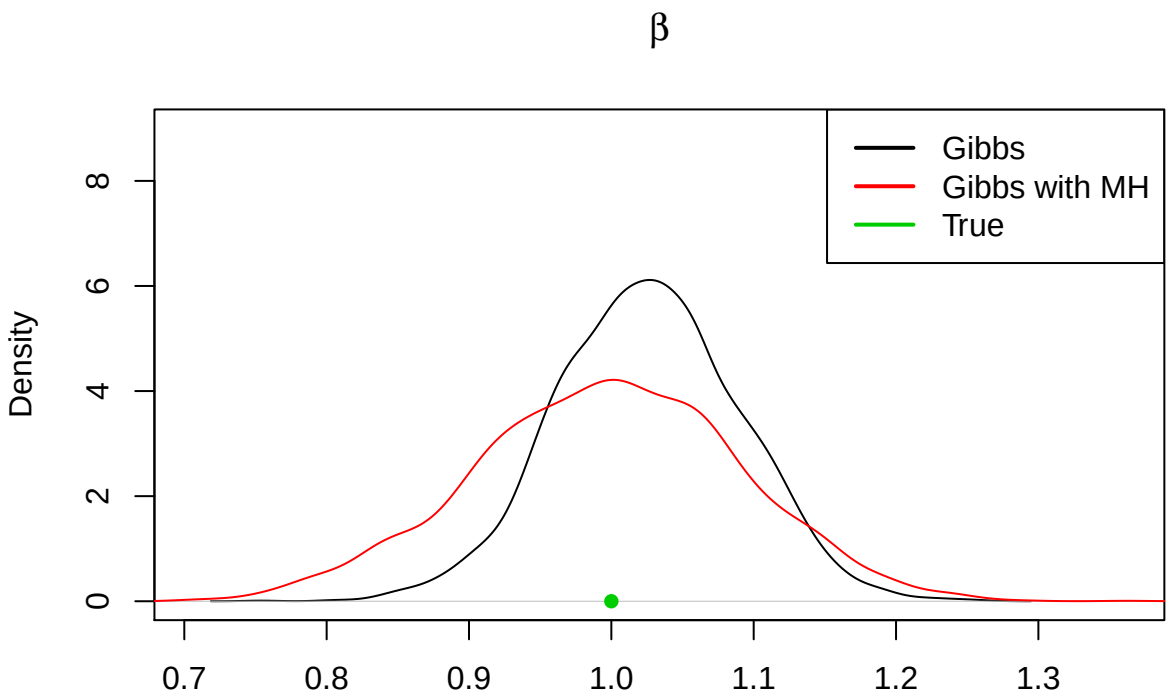


Comparing both MCMC methods (Posterior Densities)

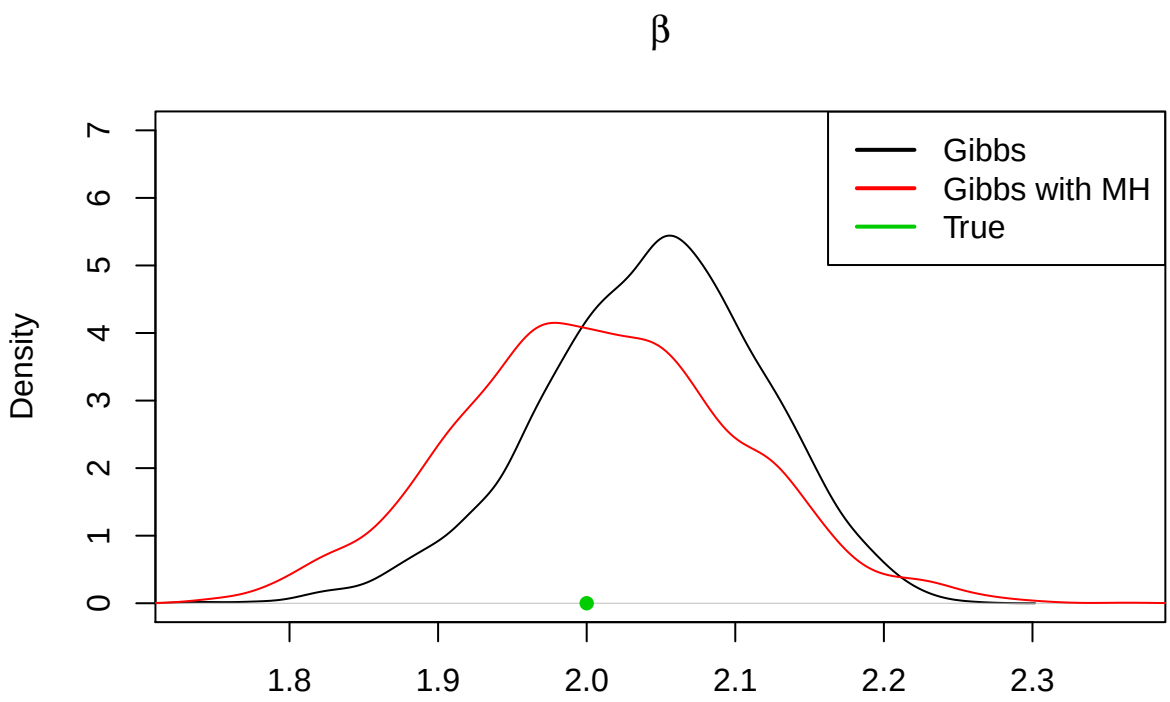
```
par(mfrow=c(1,1))
limx = range(draws1[,1],draws2[,1])
plot(density(draws1[,1]),xlim=limx,ylim=c(0,9),main=expression(beta),xlab="")
lines(density(draws2[,1]),col=2)
points(beta.t[1],0,pch=16,col=3)
legend("topright",legend=c("Gibbs","Gibbs with MH","True"),col=1:3,lty=1,lwd=2)
```



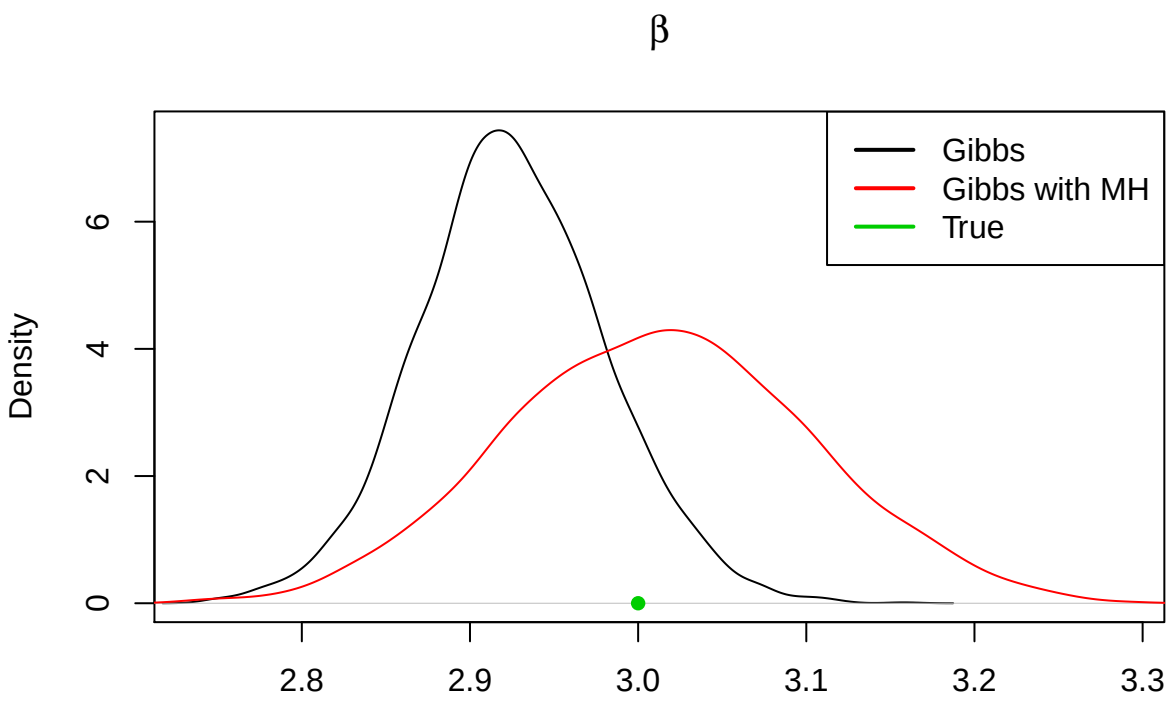

```
par(mfrow=c(1,1))
limx = range(draws1[,2],draws2[,2])
plot(density(draws1[,2]),xlim=limx,ylim=c(0,9),main=expression(beta),xlab="")
lines(density(draws2[,2]),col=2)
points(beta.t[2],0,pch=16,col=3)
legend("topright",legend=c("Gibbs","Gibbs with MH","True"),col=1:3,lty=1,lwd=2)
```



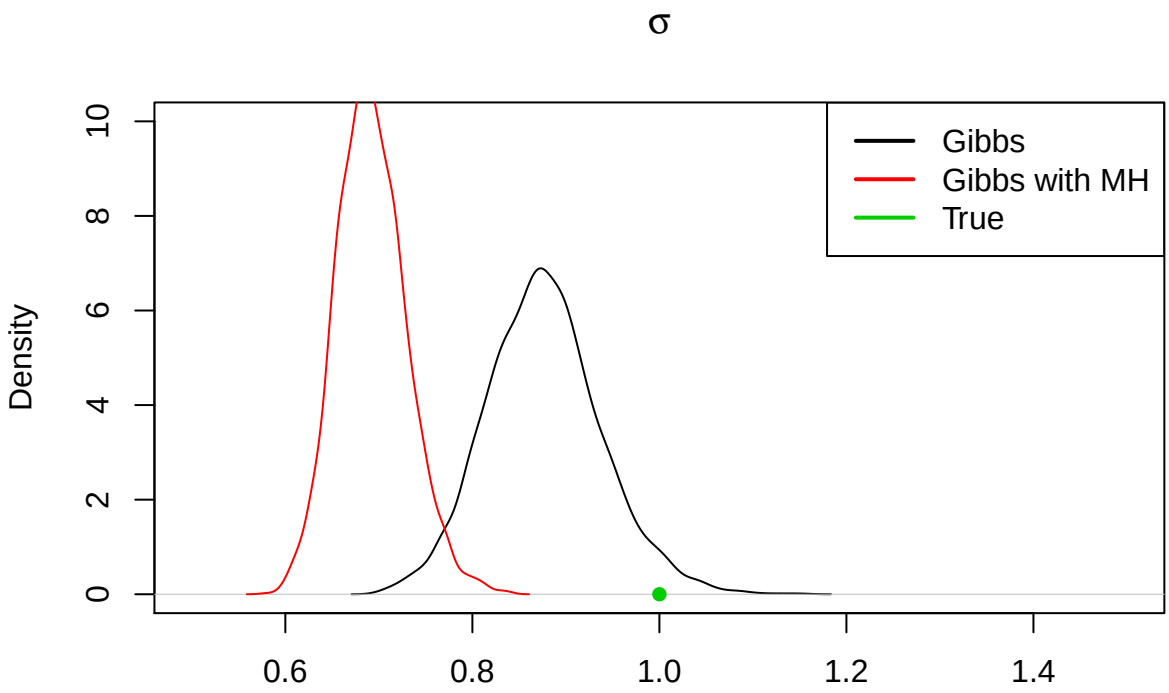

```
par(mfrow=c(1,1))
limx = range(draws1[,3],draws2[,3])
plot(density(draws1[,3]),xlim=limx,ylim=c(0,7),main=expression(beta),xlab="")
lines(density(draws2[,3]),col=2)
points(beta.t[3],0,pch=16,col=3)
legend("topright",legend=c("Gibbs","Gibbs with MH","True"),col=1:3,lty=1,lwd=2)
```




```
par(mfrow=c(1,1))
limx = range(draws1[,4],draws2[,4])
plot(density(draws1[,4]),xlim=limx,main=expression(beta),xlab="")
lines(density(draws2[,4]),col=2)
points(beta.t[4],0,pch=16,col=3)
legend("topright",legend=c("Gibbs","Gibbs with MH","True"),col=1:3,lty=1,lwd=2)
```




```
par(mfrow=c(1,1))
limx = range(c(0.5, 1.5))
plot(density(draws1[,5]),xlim=limx,ylim=c(0,10),main=expression(sigma),xlab="")
lines(density(draws2[,5]),col=2)
points(sig.t,0,pch=16,col=3)
legend("topright",legend=c("Gibbs","Gibbs with MH","True"),col=1:3,lty=1,lwd=2)
```



Note that the MH within the Gibbs generate a not so good result for the parameter σ. For this parameter, the full Gibbs is better.

Efficiency

```
# Using 'mcmcse' package

library(mcmcse)
```

```
## Warning: package 'mcmcse' was built under R version 3.5.3
```

```
## mcmcse: Monte Carlo Standard Errors for MCMC
## Version 1.4-1 created on 2020-01-29.
## copyright (c) 2012, James M. Flegal, University of California, Riverside
##                     John Hughes, University of Colorado, Denver
##                     Dootika Vats, University of Warwick
##                     Ning Dai, University of Minnesota
##  For citation information, type citation("mcmcse").
##  Type help("mcmcse-package") to get started.
```

```
gibbs.essp = round( (ess(draws1)/gibbs.time), 2)
MH.essp = round( (ess(draws2)/MH.time) , 2)
table = rbind(gibbs.essp,MH.essp)
colnames(table) = c('Beta1','Beta2','Beta3','Beta4','Sigma')
```

```
rownames(table) = c('Gibbs', 'MH')
table
```

```
##       Beta1 Beta2 Beta3 Beta4 Sigma
## Gibbs 23.31 19.90 13.33 28.49 33.76
## MH    61.29 10.37 59.43 51.06  1.56
```

This package calculates efficiency as:

$$ESS = M\frac{\lambda^2}{\sigma^2}$$

#where λ^2 is the sample variance and σ^2 is an estimate of the variance in the CLT.

In terms of this concept of sampling efficiency, it seems that the Metropolis Hastings step within the Gibbs sampler tend to generate better results for β_2,β_3 and β_4. But the full-fledged Gibbs performed better in sampling σ and β_1.

When we calculate

$$ESS = \frac{M}{1 + 2\sum_{n=1}^{\infty}\rho}$$

```
ess.Emu.gibbs1 = round(M/(1+2*abs(sum(acf(draws1[,1],lag.max=M,plot=FALSE)$acf[2:M]))) )
ess.Emu.gibbs2 = round(M/(1+2*abs(sum(acf(draws1[,2],lag.max=M,plot=FALSE)$acf[2:M]))) )
ess.Emu.gibbs3 = round(M/(1+2*abs(sum(acf(draws1[,3],lag.max=M,plot=FALSE)$acf[2:M]))) )
ess.Emu.gibbs4 = round(M/(1+2*abs(sum(acf(draws1[,4],lag.max=M,plot=FALSE)$acf[2:M]))) )
ess.Emu.gibbs5 = round(M/(1+2*abs(sum(acf(draws1[,5],lag.max=M,plot=FALSE)$acf[2:M]))) )

essps.Emu.gibbs1 = ess.Emu.gibbs1/gibbs.time
essps.Emu.gibbs2 = ess.Emu.gibbs2/gibbs.time
essps.Emu.gibbs3 = ess.Emu.gibbs3/gibbs.time
essps.Emu.gibbs4 = ess.Emu.gibbs4/gibbs.time
essps.Emu.gibbs5 = ess.Emu.gibbs5/gibbs.time




ess.Emu.MH1 = round(M/(1+2*abs(sum(acf(draws2[,1],lag.max=M,plot=FALSE)$acf[2:M]))))
ess.Emu.MH2 = round(M/(1+2*abs(sum(acf(draws2[,2],lag.max=M,plot=FALSE)$acf[2:M]))))
ess.Emu.MH3 = round(M/(1+2*abs(sum(acf(draws2[,3],lag.max=M,plot=FALSE)$acf[2:M]))))
ess.Emu.MH4 = round(M/(1+2*abs(sum(acf(draws2[,4],lag.max=M,plot=FALSE)$acf[2:M]))))
ess.Emu.MH5 = round(M/(1+2*abs(sum(acf(draws2[,5],lag.max=M,plot=FALSE)$acf[2:M]))))

essps.Emu.MH1 = ess.Emu.MH1/MH.time
essps.Emu.MH2 = ess.Emu.MH2/MH.time
essps.Emu.MH3 = ess.Emu.MH3/MH.time
essps.Emu.MH4 = ess.Emu.MH4/MH.time
```

```r
essps.Emu.MH5 = ess.Emu.MH5/MH.time



table_beta1 = cbind(c(gibbs.time,ess.Emu.gibbs1,essps.Emu.gibbs1),
              c(MH.time,ess.Emu.MH1,essps.Emu.MH1))

table_beta2 = cbind(c(gibbs.time,ess.Emu.gibbs2,essps.Emu.gibbs2),
                    c(MH.time,ess.Emu.MH2,essps.Emu.MH2))

table_beta3 = cbind(c(gibbs.time,ess.Emu.gibbs3,essps.Emu.gibbs3),
                    c(MH.time,ess.Emu.MH3,essps.Emu.MH3))

table_beta4 = cbind(c(gibbs.time,ess.Emu.gibbs4,essps.Emu.gibbs4),
                    c(MH.time,ess.Emu.MH4,essps.Emu.MH4))


table_sigma = cbind(c(gibbs.time,ess.Emu.gibbs5,essps.Emu.gibbs5),
                    c(MH.time,ess.Emu.MH5,essps.Emu.MH5))

rownames(table_beta1) = rownames(table_beta2) = rownames(table_beta3) =
  rownames(table_beta4) = rownames(table_sigma) = c("Time (sec)","ESS","ESS/sec")

colnames(table_beta1) = colnames(table_beta2) = colnames(table_beta3) =
  colnames(table_beta4) =colnames(table_sigma) =c("Gibbs","MH")
```

```r
table_beta1
```

```
##                 Gibbs         MH
## Time (sec)   31.45000   48.95000
## ESS        1500.00000 1500.00000
## ESS/sec      47.69475   30.64351
```

```r
table_beta2
```

```
##                 Gibbs         MH
## Time (sec)   31.45000   48.95000
## ESS        1500.00000 1500.00000
## ESS/sec      47.69475   30.64351
```

```r
table_beta3
```

```
##                 Gibbs         MH
## Time (sec)   31.45000   48.95000
## ESS        1500.00000 1500.00000
## ESS/sec      47.69475   30.64351
```

```r
table_beta4
```

```
##                 Gibbs         MH
## Time (sec)   31.45000   48.95000
## ESS        1500.00000 1500.00000
## ESS/sec      47.69475   30.64351
```

```r
table_sigma
```

```
##                 Gibbs         MH
```

```
## Time (sec)   31.45000   48.95000
## ESS        1500.00000 1500.00000
## ESS/sec      47.69475   30.64351
```

We can note that, of course, the results are equal among different parameters, with the full-fledged Gibbs performing better in the sampling procedure than the MH within the Gibbs. It means that the full Gibbs has lower autocorrelation in the chain, and also it avoids to cycle with a acceptance/rejection step, which saves time for the algorithm.