Statistical Computing Final Project

Wei-Chen Chang r12227118

Due: 2024-06-05

The Model

Suppose Y_{ij} are the birthweight for gender i , and assume

$$Y_{ij} \overset{ind}{\sim} N(\mu_i, \sigma^2),\ \mu_i \overset{ind}{\sim} N(\eta, \tau^2)$$

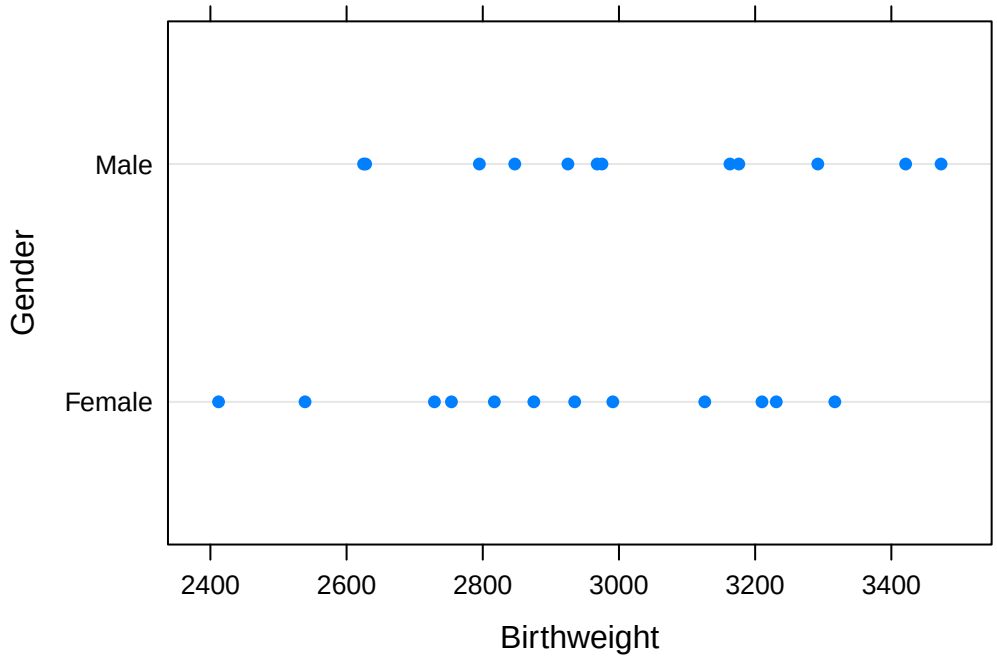
for $i = 1, 2, j = 1, ..., n_i; n = n_1 + n_2$, and prior

$$\pi(\eta, \tau^2, \sigma^2) \propto IG(\tau^2; a_\tau, b_\tau) \cdot Ca^+(\sigma^2; 0, b_\sigma)$$

Load dataset

Load dataset, take a look, and setting variables.

```
library(lattice)
library(tidyverse)
# Load Dataset
data(birthweight,package="LearnBayes")
bweight.list <- list("gender" = birthweight$gender+1,
                     "weight"= birthweight$weight,
                     "N" = nrow(birthweight),
                     "eta_mean" = mean(birthweight$weight))

mygender <- with(birthweight, ifelse(birthweight$gender==0,"Male","Female"))
lattice::dotplot(mygender ~ weight, data = birthweight,
xlab = "Birthweight", ylab = "Gender")
```




```
bw_summerize <-  birthweight|>
  group_by(gender)|>
  summarise(N=n(),
            weight.mean = mean(weight),
            weight.sd = sd(weight),
            weight.var = var(weight))
n.gender <- bw_summerize$N
ybar <- bw_summerize$weight.mean
bw_summerize
```

```
## # A tibble: 2 x 5
##   gender     N weight.mean weight.sd weight.var
##    <int> <int>       <dbl>     <dbl>      <dbl>
## 1      0    12       3024       284.     80780.
## 2      1    12       2911.      280.     78648.
```

Moreover, here compute the variance between the sample mean different gender, which is 6346.89.

1. Implementation in JAGS

Specifying Model in JAGS

Since there's no Cauchy distribution function in JAGS, we use $t(x; \mu = 0, \tau, k = 1)$ via `dt` Cauchy distribution $Ca(x; 0, b)$. Note that in such parameterization in JAGS, $\tau = \frac{1}{b^2}$.

The hyperparameters b_τ, b_σ in Cauchy priors of τ^2, σ^2 were set as 10,000 and 100,000 because the sample variance from the data is large. Moreover, here we let η follows a normal hyperprior with $\mu_\eta = \frac{1}{(n_1+n_2)} \sum_{i=1}^{2} \sum_{j=1}^{n_i} y_i$, $\sigma_\eta^2 = 10000$.

Lastly, the intial values for 5 parameters were generated by the distribution below:

$$\mu_i, \eta \sim N(3000, \sigma^2 = 500^2)$$
$$\tau^2 \sim gamma(a = 10, b = 50)$$
$$\sigma^2 \sim gamma(a = 10, b = 500)$$

```
# JAGS model
modelstring ="
model{
  for(subID in 1:N){
    weight[subID] ~ dnorm(mu[gender[subID]], preci[1])
  }
  for (i in 1:2){
    mu[i] ~ dnorm(eta, preci[2])
  }
  preci[1] <- 1/sigma.sq
  preci[2] <- 1/tau.sq

## priors ##
  eta ~ dnorm(eta_mean, 1/10000)
  tau.sq ~ dt(0, t.tau_tau.sq, 1) T(0,)
  t.tau_tau.sq <- 1/b.tau^2

  sigma.sq ~ dt(0, t.tau_sigma.sq, 1) T(0,)
  t.tau_sigma.sq <- 1/b.sigma^2
  b.tau <-  10000
  b.sigma <- 100000
}
"
writeLines(modelstring, con="bweight.bug")
```

Run jags model:

```
library(coda)
library(rjags)
library(R2jags)
param.names <- c("mu", "eta", "tau.sq","sigma.sq")
# set initial values

set.seed(99)
bayes.mod.inits <- function(tau.scale=50, sigma.scale=500){
 list("mu" = rnorm(2, 3000, 500),
      "eta" = rnorm(1, 3000, 500),
      "tau.sq" = rgamma(1, 10, scale=tau.scale),
      "sigma.sq" = rgamma(1, 10, scale=sigma.scale)
      )
}
inits <- list(bayes.mod.inits(), bayes.mod.inits(), bayes.mod.inits())
bweight.jags<- jags(data = bweight.list,
     inits = inits,
     parameters.to.save = param.names,
```

```
    n.chains=3, n.iter=10000, n.burnin=1000,
    model.file = "bweight.bug")
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 24
##    Unobserved stochastic nodes: 5
##    Total graph size: 67
##
## Initializing model
```

```
bweight.jags |> print()
```

```
## Inference for Bugs model at "bweight.bug", fit using jags,
##  3 chains, each with 10000 iterations (first 1000 discarded), n.thin = 9
##  n.sims = 3000 iterations saved
##             mu.vect   sd.vect      2.5%       25%       50%       75%      97.5%
## eta        2969.416    65.973  2839.877  2927.311  2967.863  3012.020   3102.359
## mu[1]      2992.448    67.193  2863.083  2947.821  2991.696  3033.767   3136.606
## mu[2]      2942.202    68.139  2806.030  2897.131  2943.230  2987.103   3074.804
## sigma.sq 85776.879 25500.291 48864.449 67640.029 81698.020 99243.507 147651.775
## tau.sq   10323.165 14336.456   206.803  2757.143  6099.482 12711.405  46238.075
## deviance   339.506     1.965   337.095   338.048   339.046   340.409    344.523
##           Rhat n.eff
## eta      1.001  3000
## mu[1]    1.001  3000
## mu[2]    1.002  1300
## sigma.sq 1.002  1800
## tau.sq   1.002  1900
## deviance 1.002  1400
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 1.9 and DIC = 341.4
## DIC is an estimate of expected predictive error (lower deviance is better).
```

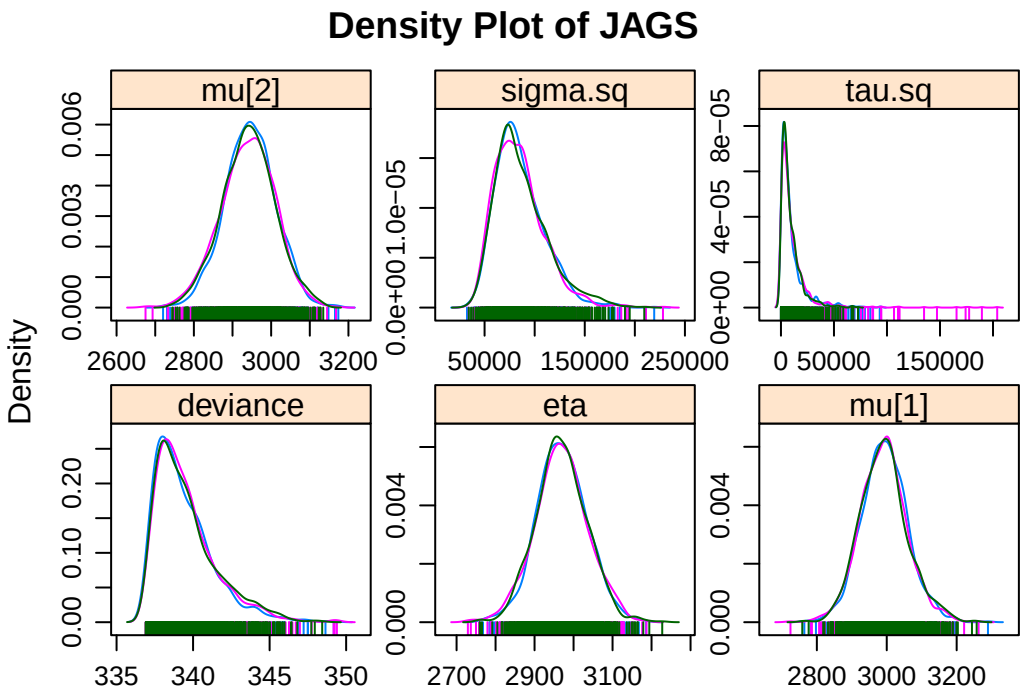
For visulaizing purpose, the posterior density plot and other diagnositic plots were shown below.

```
jags.mcmc <- as.mcmc(bweight.jags)
gelman.diag(jags.mcmc)
```

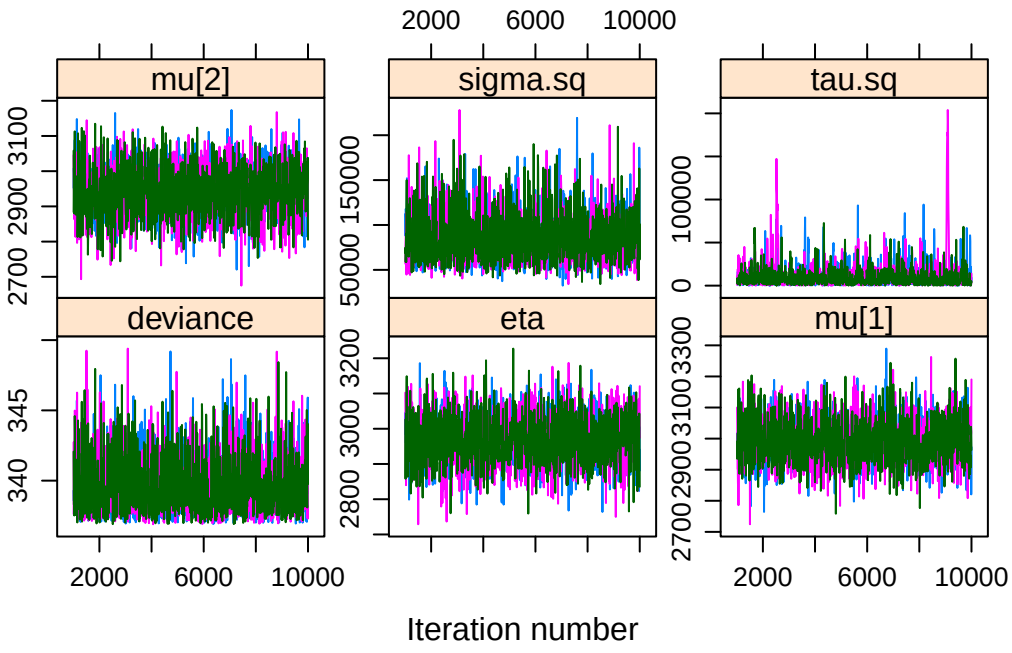
```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## deviance      0.999       1.00
## eta           1.000       1.00
## mu[1]         1.000       1.00
```

```
## mu[2]          1.001       1.00
## sigma.sq       1.003       1.01
## tau.sq         1.088       1.12
##
## Multivariate psrf
##
## 1.01
```

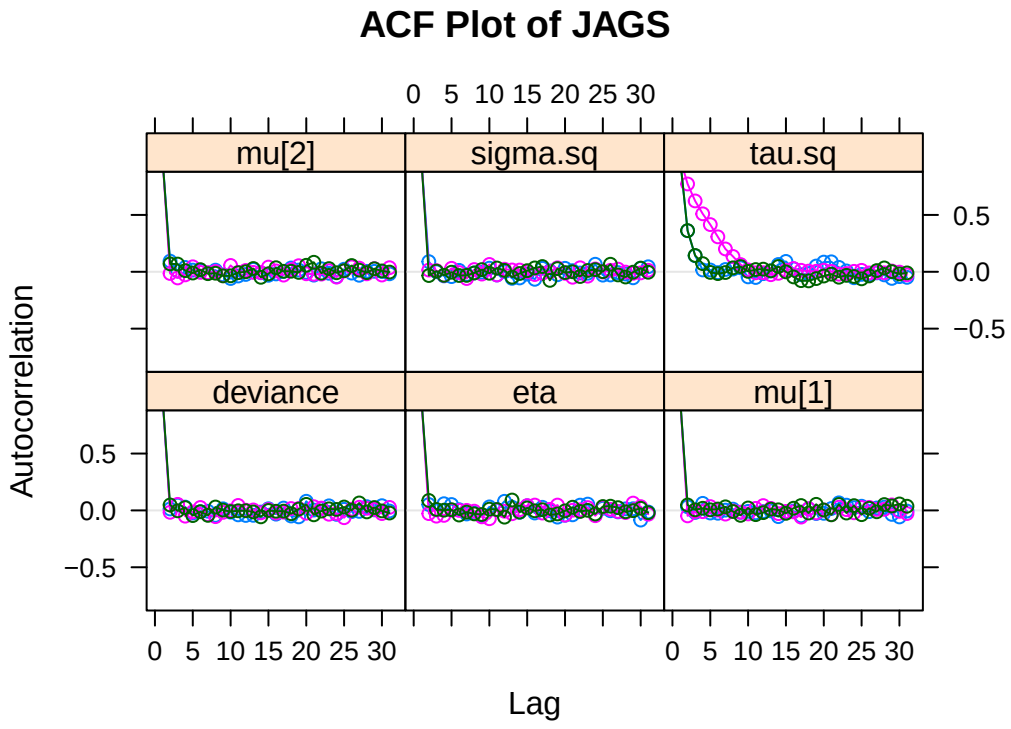
```
densityplot(jags.mcmc, layout=c(3,2), aspect="fill",
       main = "Density Plot of JAGS")
```




```
xyplot(jags.mcmc, layout=c(3,2), aspect="fill",
       main = "Divergence Plot of JAGS")
```




```
acfplot(jags.mcmc, main = "ACF Plot of JAGS", layout=c(3,2), aspect="fill")
```



Comments

All parameters seems converge ok from the density plot and caterpillar plot, but τ^2 seem suffer from auto-correlation between samples (See ACF plot).

The difference between posterior mean of μ_1, μ_2 smaller than difference of sample mean. And σ^2 and τ^2 is much larger than sample estimates.

2-4.

To make life easier, A Gibbs sampler function were written to implement different sampling methods for generating σ.

Gibbs Sampler functions

To achieve such goal, first all other sampler from full conditional distribution must be specified first.

To align the result from JAGS, for the following MCMC sequences, there are 10000 iterations for each, first 1000 samples were burnt-in, and thinning 9 samples, Resulting a 1000 valid samples for each chain. And each sampling method runs 3 chains.

Lastly, the value of hyperparameters were assigned as below: $a_\tau = 1, b_\tau = 10000$ in Inverse-Gamma priors, and $b_\sigma = 10000$. in truncated Cauchy priors

```
library(nimble) # help generate inverted gamma
# ---- MCMC Generators ----
mu_gen <- function(i, eta, sigma.sq, tau.sq){
  if (!(i %in% 1:2)){
    return("invalid i")}
  Nor.mu <- (1/(sigma.sq/n.gender[i]) + 1/tau.sq)^(-1) * (ybar[i]/(sigma.sq/n.gender[i]) + eta/tau.sq)
  Nor.sig_sq <- (1/(sigma.sq/n.gender[i]) + 1/tau.sq)^(-1)
  rnorm(1, Nor.mu, sqrt(Nor.sig_sq))
}

eta_gen <- function(mu, # 2*1 vector
                           tau.sq) {
  mu_bar <-  mean(mu)
  rnorm(1, mu_bar, sqrt(tau.sq/2))
}

tau.sq_gen <- function(mu, # 2*1 vector
                       eta, a.tau, b.tau){
  shape <-  a.tau+1
  scale <-  mean((mu-eta)^2) + b.tau
  rinvgamma(1, shape = shape, scale = scale) # from `nimble`
}

sigma.sq_gen <- function(mu, b.sigma, last_sigma.sq,
    type=c("AR", "MH.ind","MH"), y=birthweight){
  type <- match.arg(type)
  a <- (nrow(y)/2)-1
  b <- ((sum(y[y$gender==0,"weight"] - mu[1])^2) + (sum(y[y$gender==1,"weight"] - mu[2])^2))/2
  Accept <- FALSE
```

```
  while (!Accept) {
    sigma.sq <- ifelse(type == "MH",
                       rchisq(1, df = last_sigma.sq),
                       rinvgamma(1, shape = a, scale = b))

    if (type == "AR"){ #A-R sampling
      c_max <- dcauchy(0, location = 0, scale = b.sigma)
      accept_rate <-
        dcauchy(sigma.sq, 0, b.sigma)/c_max
      # truncaction constant were cancel out
    }
    else if (type == "MH.ind"){ #MH independent
      prop <- dcauchy(sigma.sq, 0, b.sigma)/dcauchy(last_sigma.sq, 0, b.sigma)
      accept_rate <- min(1, prop)
    }
    else if(type == "MH"){
      q_func <- function(x){
        dinvgamma(x, shape = a, scale = b)*2*dcauchy(x, scale = b.sigma)*(x>0)
      }
      num <- q_func(sigma.sq) * dchisq(last_sigma.sq, df = sigma.sq)
      deno <- q_func(last_sigma.sq) * dchisq(sigma.sq, df = last_sigma.sq)
      accept_rate <- ifelse(deno==0, 1 , min((num/deno), 1))
    }
    u <- runif(1)
    Accept <- (u < accept_rate)
  }
  return(sigma.sq)
}

# ---- The Gibbs-Sampler function ----
gibbs <- function(inits, Nsim, type = c("AR", "MH.ind","MH"),
                  hyp.param=list(a.tau=1,b.tau=10000,b.sigma=10000),
                  burnin=1000, thining=9, chain=3){
  type <- match.arg(type)
  MCMC_chain <- function(init){
    MCMC.mtx <-
      matrix(nrow = Nsim, ncol = 5,
             dimnames =
               list(c(), c("mu1", "mu2", "eta","sigma.sq","tau.sq"))
             )
    mu1 <- init$mu[1]
    mu2 <- init$mu[2]
    eta <- init$eta
    sigma.sq <- init$sigma.sq
    tau.sq <- init$tau.sq
    MCMC.mtx[1, ] <- c(mu1, mu2, eta, sigma.sq, tau.sq)
    # hyperparameters
    a.tau <- hyp.param$a.tau # larger a, Expectation smaller
    b.tau <- hyp.param$b.tau
    b.sigma <- hyp.param$b.sigma
    for (iter in 2:Nsim){
      mu1 <- mu_gen(1, eta, sigma.sq, tau.sq)
      mu2 <- mu_gen(2, eta, sigma.sq, tau.sq)
```

```
      eta <- eta_gen(mu=c(mu1, mu2), tau.sq)
      tau.sq <- tau.sq_gen(mu = c(mu1, mu2), eta = eta,
                           a.tau = a.tau, b.tau = b.tau)
      sigma.sq <- sigma.sq_gen(mu = c(mu1, mu2), b.sigma=b.sigma,
                               last_sigma.sq = sigma.sq, type=type)
      # print(MCMC.mtx)
      MCMC.mtx[iter, ] <- c(mu1, mu2, eta, sigma.sq, tau.sq)
    }
    MCMC.mtx <- MCMC.mtx[seq(1+burnin, Nsim, by=thining),]
    mcmc(MCMC.mtx, start = burnin+thining, end = Nsim, thin = thining)
  }
  for (i in 1:chain){
    if (i==1){
      MCMC_list <- list(MCMC_chain(inits[[i]]))
    }
    else{
      MCMC_list[[i]] <- MCMC_chain(inits[[i]])
    }
  }
  mcmc.list(MCMC_list)
  # return(mcmc.list(MCMC_list))
}
```

2. AR method

Acceptance-Reject sampling with $\sigma^* \sim IG(a, b)$ and thus $C_{max} = Ca^+(0; 0, b_\sigma)$ and the acceptance probability is $Ca^+(\sigma^*; 0, b_\sigma)/C_{max}$.

Note that $a = \frac{n}{2} - 1$, $b = \sum_{i=2}^{2} \sum_{j=1}^{n_i} (y_{ij} - \mu_i)^2/2$.

Here the starting values were set as following:

```
set.seed(99)
inits <- list(bayes.mod.inits(tau.scale=50), bayes.mod.inits(tau.scale=50), bayes.mod.inits(tau.scale=5
AR.mcmc <- gibbs(inits, Nsim=10000,
                 burnin=1000, thining=9, chain=3,
                 type = "AR")#,chain=1)
# resluts
summary(AR.mcmc)
```

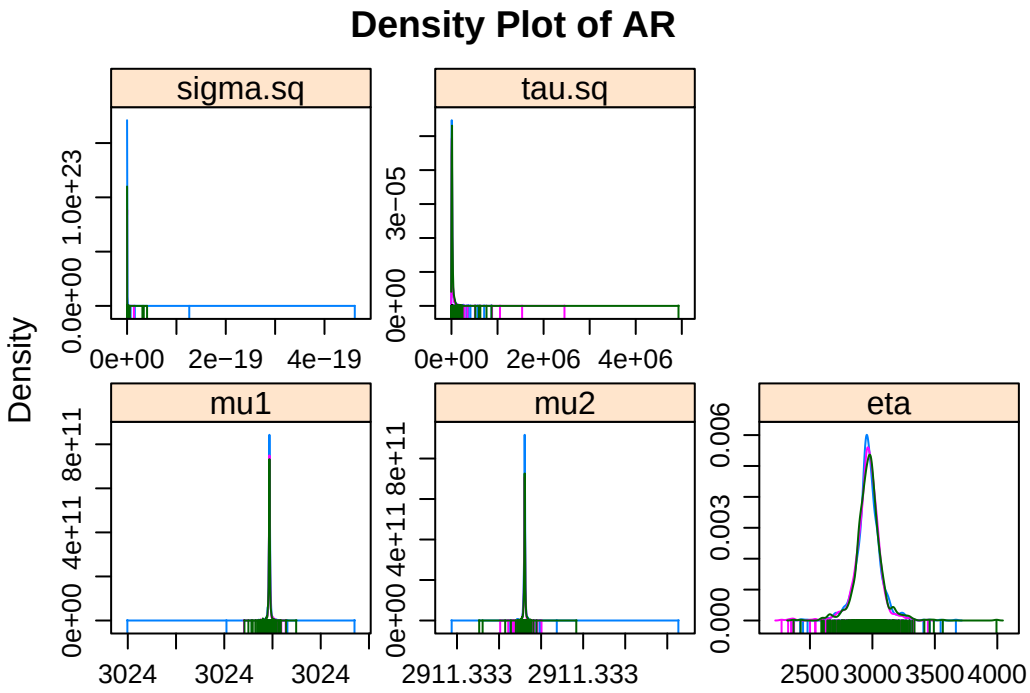
```
##
## Iterations = 1009:10000
## Thinning interval = 9
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean        SD  Naive SE Time-series SE
## mu1      3.024e+03 3.803e-12 6.943e-14          0.000
## mu2      2.911e+03 4.568e-12 8.340e-14          0.000
```

```
## eta      2.967e+03 1.062e+02 1.940e+00          1.993
## sigma.sq 2.943e-22 8.817e-21 1.610e-22          0.000
## tau.sq   2.587e+04 1.171e+05 2.137e+03       2137.640
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## mu1      3.024e+03 3.024e+03 3.024e+03 3.024e+03 3.024e+03
## mu2      2.911e+03 2.911e+03 2.911e+03 2.911e+03 2.911e+03
## eta      2.751e+03 2.919e+03 2.967e+03 3.017e+03 3.183e+03
## sigma.sq 1.082e-25 8.706e-25 3.305e-24 1.310e-23 3.720e-22
## tau.sq   2.762e+03 6.451e+03 1.123e+04 2.080e+04 1.056e+05
```

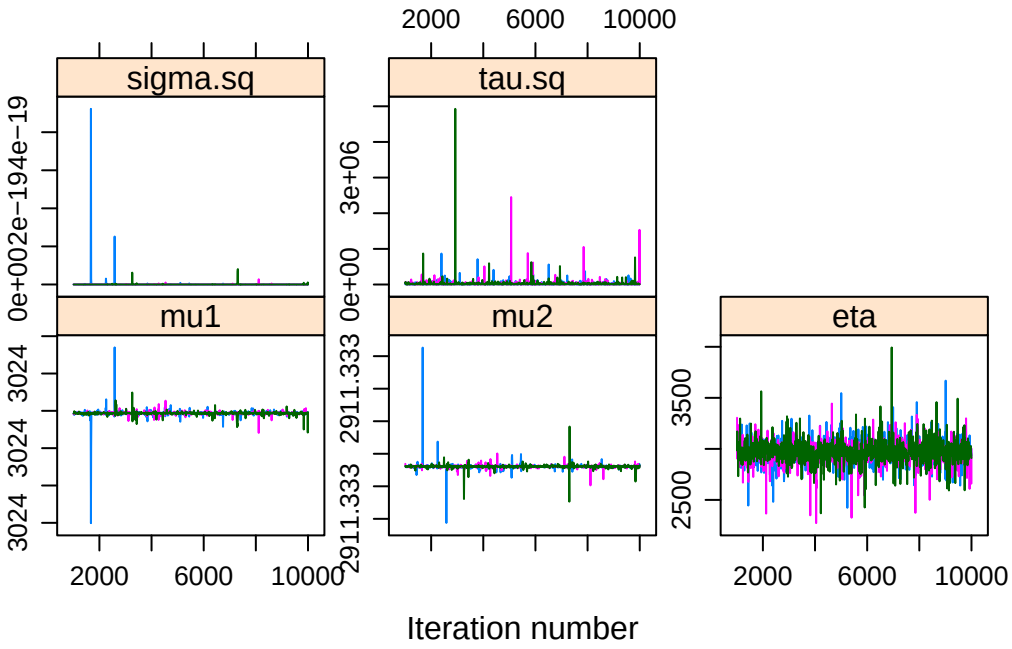
```
gelman.diag(AR.mcmc)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## mu1            1.76       1.86
## mu2            1.10       1.10
## eta            1.00       1.01
## sigma.sq       1.25       1.31
## tau.sq         1.18       1.20
##
## Multivariate psrf
##
## 1.02
```

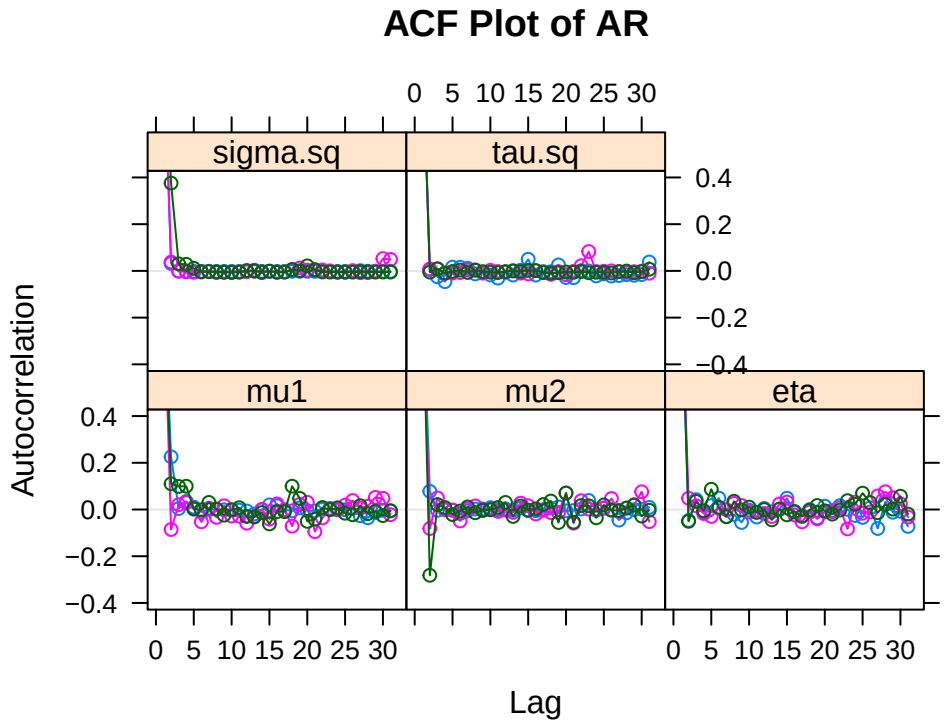
```
densityplot(AR.mcmc, layout=c(3,2), aspect="fill",
       main = "Density Plot of AR")
```




```
xyplot(AR.mcmc, layout=c(3,2), aspect="fill",
       main = "Divergence Plot of AR")
```



```
acfplot(AR.mcmc, main = "ACF Plot of AR", layout=c(3,2), aspect="fill")
```



Comments

The convergence of parameters seems to be problematic in Rubin-Gelman statistics (Potential scale reduction factors), especially for μ_1. But From the density plot and the caterpillar plot one can find the variation of estimates were surprisingly very small. The empirical s.d. of μ_1, μ_2 is even about 10^{-12}!

The posterior mean of μ_1, μ_2 is identical to sample estimates. But σ^2 is extremely close to 0 (about 10^{-20}), which is unreasonable compare to the real data. The value of τ^2 is quite reasonable, though.

3. independence M-H sampling

Independence Metropolis-Hastings with $\sigma^* \sim IG(a, b)$ and thus the acceptance probability is

$$\min\{Ca^+(\sigma^*; 0, b_\sigma)/Ca^+(\sigma^{(t)}; 0, b_\sigma), 1\}$$

.

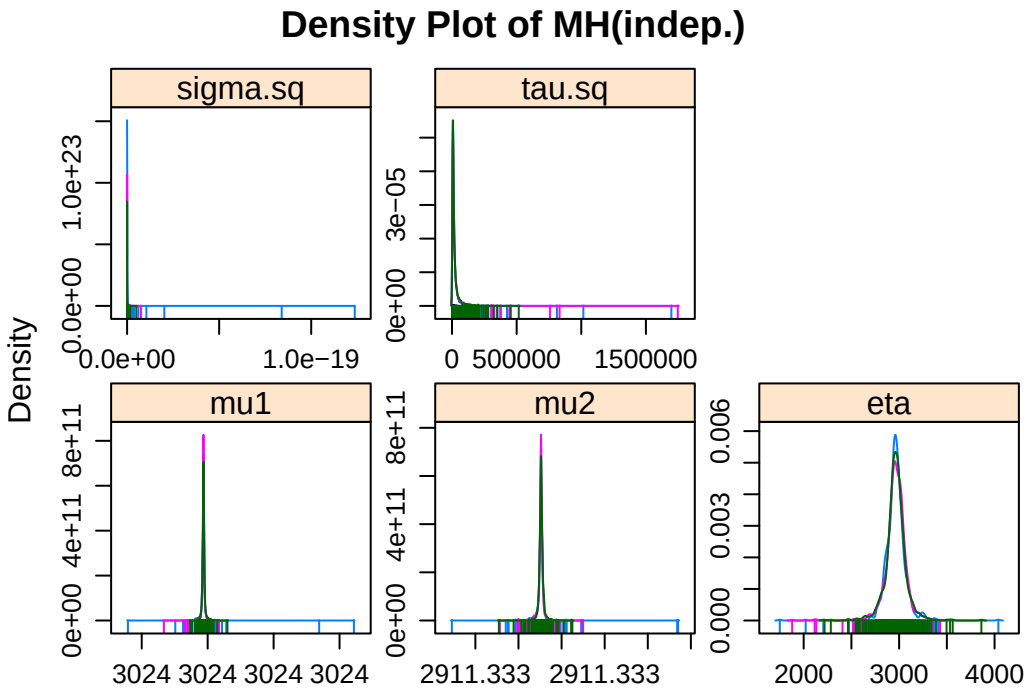
```
set.seed(99)
inits <- list(bayes.mod.inits(), bayes.mod.inits(), bayes.mod.inits())
MH.ind.mcmc <- gibbs(inits,
            Nsim=10000, burnin=1000, thining=9, chain=3,
            type = "MH.ind")

summary(MH.ind.mcmc)
```

```
##
## Iterations = 1009:10000
## Thinning interval = 9
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean        SD  Naive SE Time-series SE
## mu1      3.024e+03 3.312e-12 6.047e-14          0.000
## mu2      2.911e+03 2.542e-12 4.642e-14          0.000
## eta      2.964e+03 1.132e+02 2.067e+00          2.106
## sigma.sq 1.209e-22 2.771e-21 5.059e-23          0.000
## tau.sq   2.393e+04 6.461e+04 1.180e+03       1189.983
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## mu1      3.024e+03 3.024e+03 3.024e+03 3.024e+03 3.024e+03
## mu2      2.911e+03 2.911e+03 2.911e+03 2.911e+03 2.911e+03
## eta      2.758e+03 2.916e+03 2.965e+03 3.017e+03 3.181e+03
## sigma.sq 1.125e-25 7.137e-25 3.093e-24 1.166e-23 3.385e-22
## tau.sq   2.887e+03 6.451e+03 1.104e+04 2.180e+04 1.187e+05
```

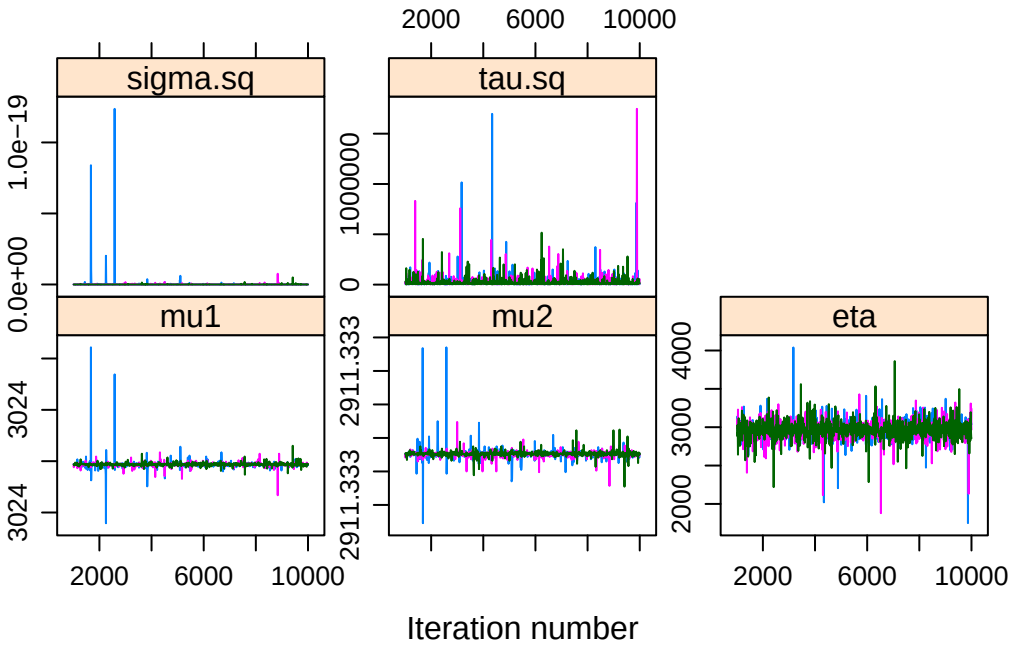
```
gelman.diag(MH.ind.mcmc)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## mu1           1.005       1.00
## mu2           1.009       1.01
## eta           0.999       1.00
## sigma.sq      1.015       1.02
## tau.sq        1.090       1.09
##
## Multivariate psrf
##
## 1
```

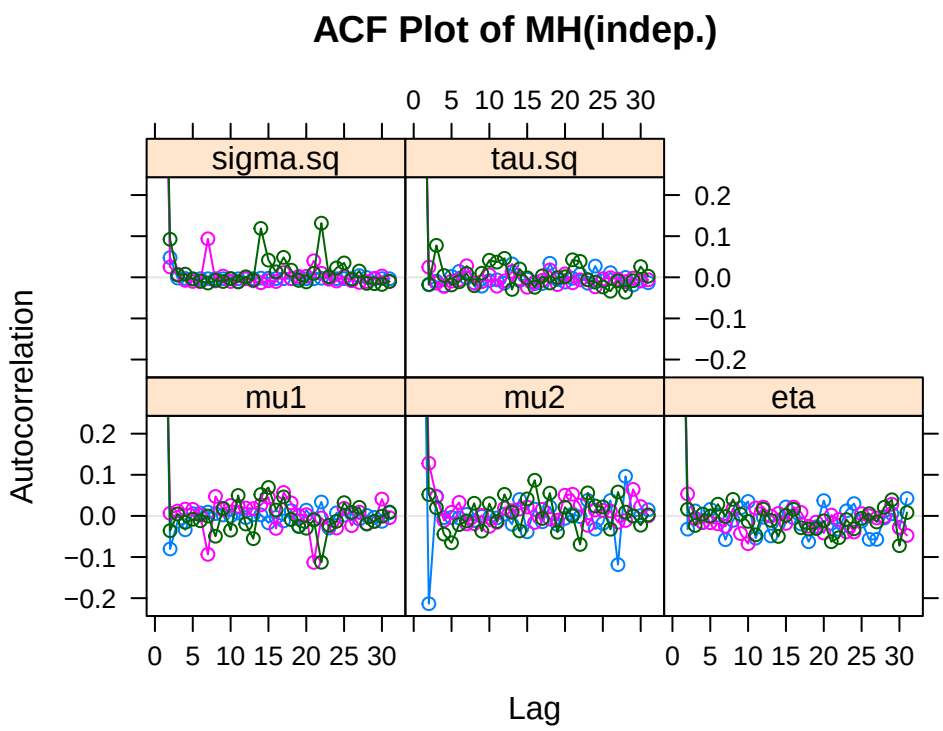
```
densityplot(MH.ind.mcmc, layout=c(3,2), aspect="fill",
       main = "Density Plot of MH(indep.)")
```




```
xyplot(MH.ind.mcmc, layout=c(3,2), aspect="fill",
       main = "Divergence Plot of MH(indep.)")
```



```
acfplot(MH.ind.mcmc, main = "ACF Plot of MH(indep.)", layout=c(3,2), aspect="fill")
```



Comments

The results of estimation is similar to AR method. Though the G-R statistic looks better it just the magic of seeds.

Notice that the ACF of M-H (independence sampling) is more volatile than AR. It's reasonable because the MH algorithm follows Markov-Chain. Moreover, MH seems to be faster and more tolerable to different initial values.

4. Metroplis-Hastings with $\sigma^* \sim \chi^2(\cdot|\sigma^{(t)})$

The acceptance probability is

$$\min\{\frac{q(\sigma^*)\cdot\chi^2(\sigma^{(t)}|\sigma^*)}{q(\sigma^{(t)})\cdot\chi^2(\sigma^*|\sigma^{(t)})}, 1\},$$

where $q(\sigma) = IG(\sigma^2; a, b)$ůCa + $(\sigma^2; 0, b_\sigma)$.

Sampling heavily relys on the value of hyperparameters. To make the sampler works, the hyperparameters in current Metroplis-Hastings algorithm were changed as below: $a_\tau = 1, b_\tau = 100$ in Inverse-Gamma priors, and $b_\sigma = 1000$ in truncated Cauchy priors.

```
set.seed(99)
inits <- list(bayes.mod.inits(), bayes.mod.inits(), bayes.mod.inits())
hyperparams <- list(a.tau =1, b.tau = 100, b.sigma=1000)
MH.mcmc <- gibbs(inits, hyp.param = hyperparams,
```

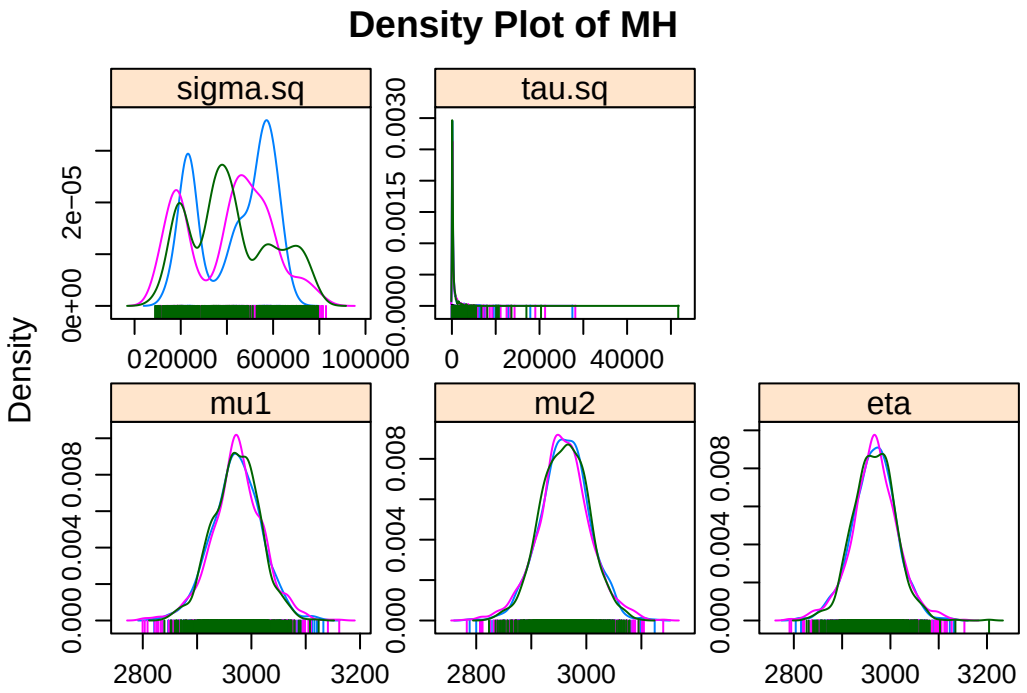
```
            Nsim = 10000, burnin=1000, thining=9, chain=3,
            type = "MH")
## results
summary(MH.mcmc)
```

```
##
## Iterations = 1009:10000
## Thinning interval = 9
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean       SD Naive SE Time-series SE
## mu1       2973.2    45.41   0.8291          1.579
## mu2       2960.4    45.06   0.8227          1.532
## eta       2967.0    45.85   0.8371          1.548
## sigma.sq 41882.5 17478.29 319.1085       7370.307
## tau.sq     555.5  1792.55  32.7274         32.730
##
## 2. Quantiles for each variable:
##
##              2.5%      25%     50%     75% 97.5%
## mu1       2879.89  2944.69  2973.2  3002.3  3064
## mu2       2871.32  2932.04  2960.7  2989.4  3049
## eta       2874.35  2938.41  2967.2  2995.8  3058
## sigma.sq 13588.52 23876.91 43252.4 56278.0 74008
## tau.sq      26.92    74.06   155.2   393.5  3633
```

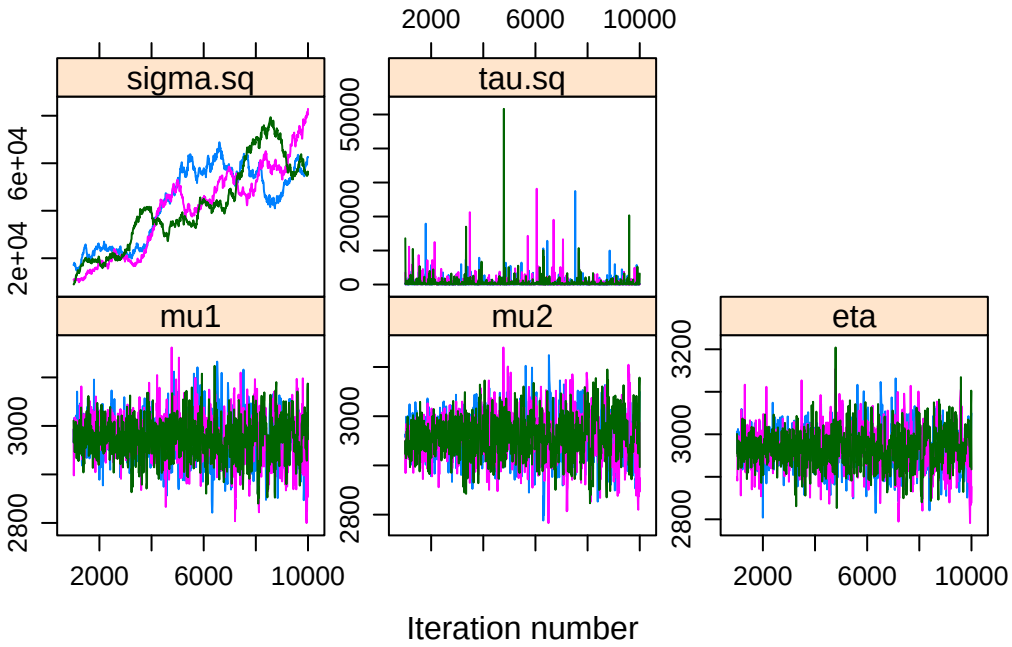
```
gelman.diag(MH.mcmc)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## mu1            1.00       1.00
## mu2            1.00       1.01
## eta            1.00       1.01
## sigma.sq       1.08       1.11
## tau.sq         1.02       1.02
##
## Multivariate psrf
##
## 1.01
```

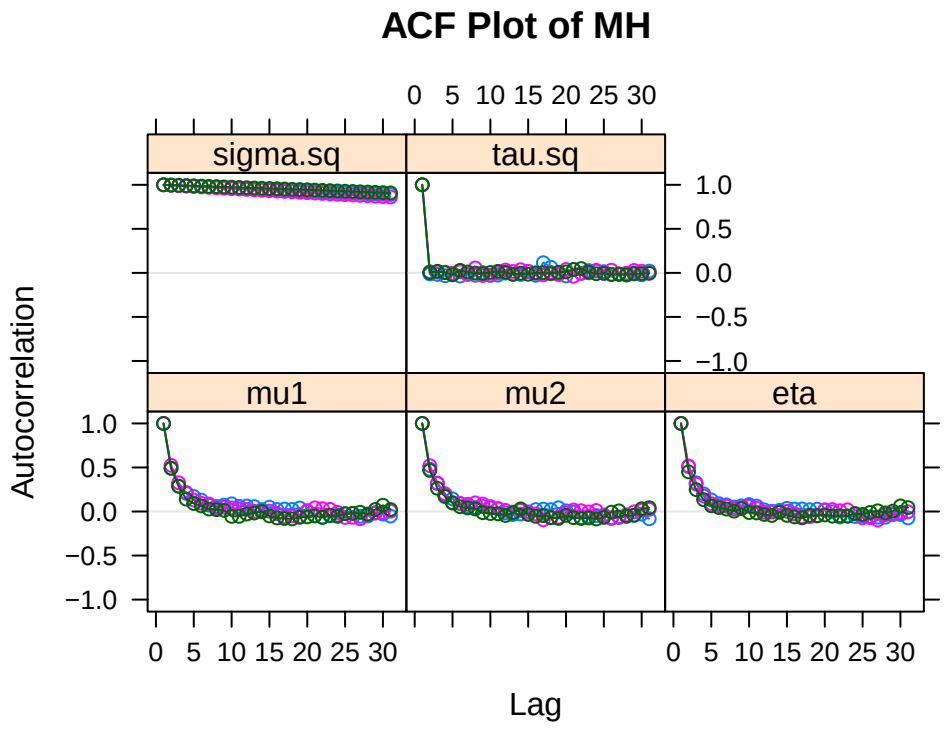
```
densityplot(MH.mcmc, layout=c(3,2), aspect="fill",
       main = "Density Plot of MH")
```




```
xyplot(MH.mcmc, layout=c(3,2), aspect="fill",
       main = "Divergence Plot of MH")
```



```
acfplot(MH.mcmc, main = "ACF Plot of MH", layout=c(3,2), aspect="fill")
```



Comments

σ^2 is not converged, and suffers from autocorrelation between samples. One can see clearly from caterpillar plot. Maybe more sampling could fix this problem. Overall, the estimates(posterior) were alike to JAGS, but with much smaller τ^2 and also difference between μ_1, μ_2.

Summary and Short Discussion

Put convergence problem aside, one can see JAGS results were similar to MH, and independence MH is similar to AR.

From the results above one can notice that if the posterior mean of μ_1, μ_2 closer to sample mean (and with smaller empirical s.d.), smaller the σ^2 is and thus more deviate from its sample estimate. It seems to be a tradeoff.